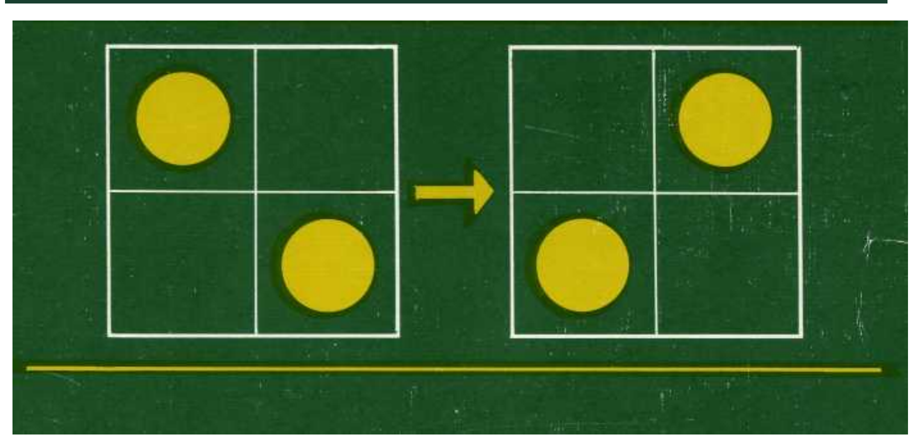
Т. Тоффоли, Н. Марголус

МАШИНЫ КЛЕТОЧНЫХ АВТОМАТОВ



Издательство «Мир»

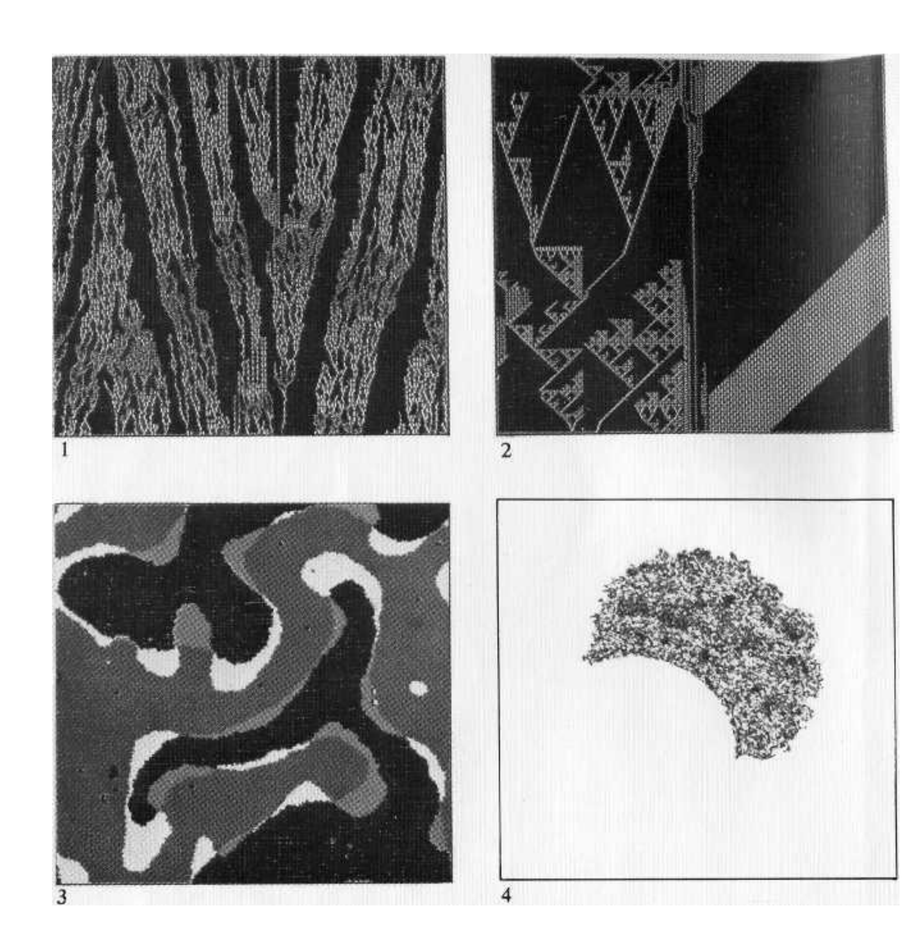


Фото 1, 2. Эволюция по правилу HGLASS (разд. 4.1) через несколько сотен шагов, если начать соответственно со случайной конфигурации и с небольшого участка нулей.

Фото 3. Поверхностное натяжение (разд. 5.4). Светлосерые области — это заливы, заполненные спустя несколько сотен шагов, темносерые — мысы, подвергшиеся размыванию.

Фото 4. Дифференциальное развитие (разд. 9.2). Мы сравниваем в реальном времени развитие двух копий одной и той же системы, начальные состояния которых отличались на один бит. Область, в которой эти две истории развития перестали совпадать (выделена на экране цветом), быстро распространяется, но не может пересечь круговой барьер.

МАШИНЫ КЛЕТОЧНЫХ АВТОМАТОВ

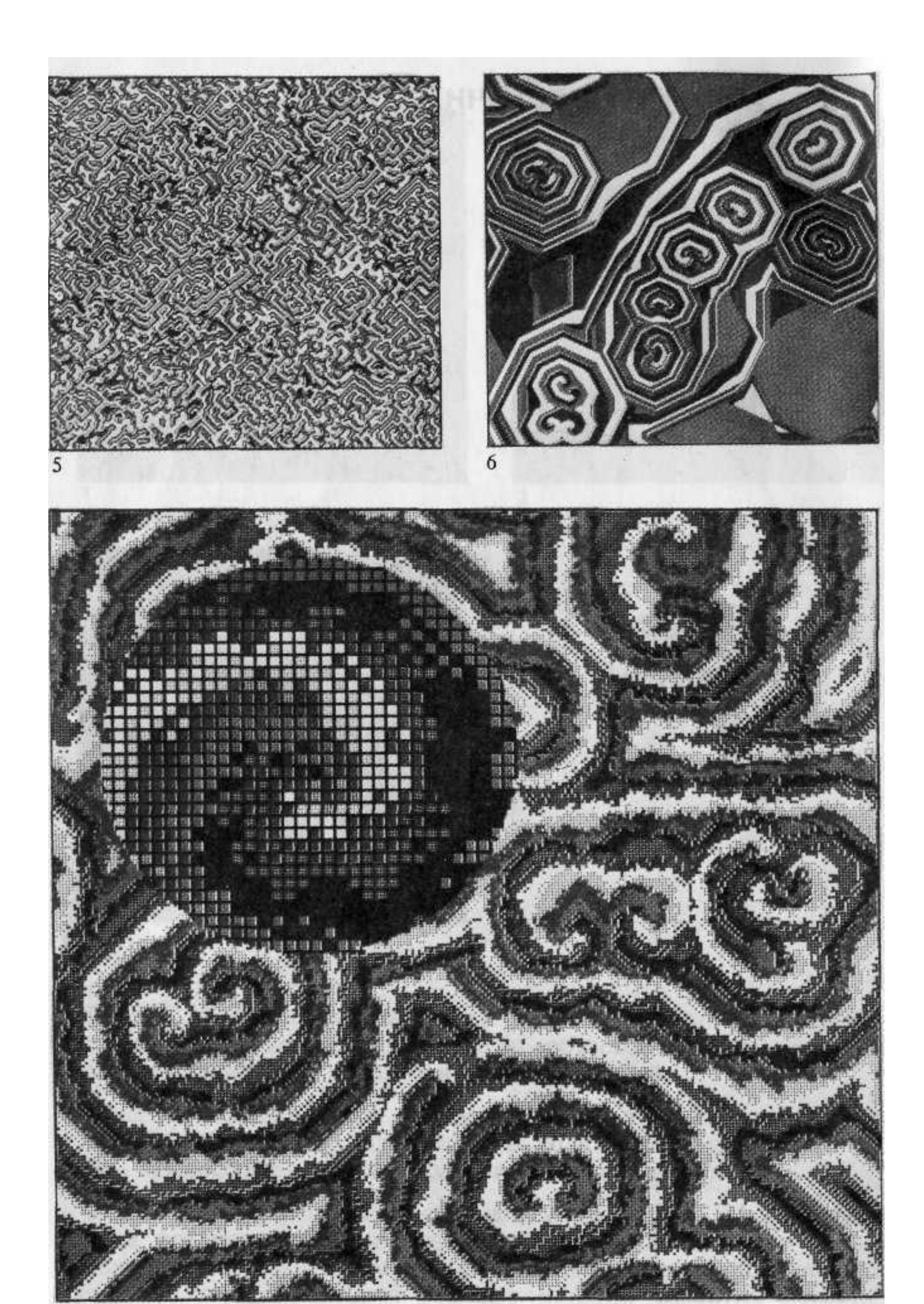
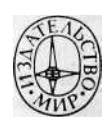


Фото 5, 6, 7. Фазовые волны в активной среде с запаздыванием при разном выборе параметров возбуждения (разд. 9.3). Система на фото 7 (где небольшой фрагмент показан увеличенным) служит моделью реакции Белоусова - Жаботинского.

Т. Тоффоли, Н. Марголус

МАШИНЫ КЛЕТОЧНЫХ *АВТОМАТОВ*

Перевод с английского
П.А. Власова и Н.В. Барабанова
под редакцией
Б. В. Баталова



Москва «МИР» 1991

ББК 32.97
Т63
УДК 519.687

Тоффоли Т., Марголус Н.
Т63 Машины клеточных автоматов: Пер. с англ. - М.: Мир, 1991. - 280 с, ил.

ISBN 5-03-001619-8

Книга американских специалистов, излагающая теорию клеточных автоматов Дж. фон Неймана и описание машины клеточных автоматов на базе персональной ЭВМ (IBM-PC). Такие машины могут использоваться для моделирования физических процессов, при решении комбинаторных и вычислительных задач, задач прикладной кибернетики. Изложение отличается простотой и ясностью и рассчитано на первоначальное ознакомление с предметом.

Для математиков-прикладников, специалистов по теории автоматов, физиков разных специальностей, аспирантов и студентов университетов.

Т $\frac{1602120000-277}{041(01)-91}$ 30-90 ББК 32.97

Редакция литературы по математическим наукам

ISBN 5-03-001619-8 (русск.)
ISBN 0-262-20060-0 (англ.)

ПРЕДИСЛОВИЕ РЕДАКТОРА ПЕРЕВОДА

Идея клеточных автоматов была сформулирована независимо Дж. фон Нейманом и К. Цусе в конце 40-х годов. Оба рассматривали их как универсальную вычислительную среду для построения алгоритмов, эквивалентную по своим выразительным возможностям машине Тьюринга. Эта идея породила волну многочисленных теоретических и прикладных исследований. Прежде всего это касается работ по созданию формальных моделей и алгоритмов на основе локальных взаимодействий, универсальных клеточных процессоров и нейрокомпьютеров. Начиная с 1976 г. в Берлине регулярно проводятся международные конференции по параллельной обработке информации на клеточных автоматах. Современный интерес к ним усиливается возможностью реализации на СБИС с высокой степенью интеграции, перспективами обработки информации на молекулярном уровне.

Книгу "Машины клеточных автоматов", перевод которой предлагается читателям, выгодно отличают широта охвата проблемы, доступность изложения и образность подачи материала. Вначале вводятся основные понятия, даются необходимые исторические сведения и описывается машина клеточных автоматов САМ-6, созданная в Массачусетском технологическом институте. Во второй части описываются различные классы правил, окрестностей и особенности динамики клеточных конфигураций. Начиная с простейших типов взаимодействий клеток, авторы постепенно подводят читателя к сложным вопросам развития клеточных структур и зависимости динамики конфигураций от пространственных и временных фаз жизни клеток. Третья часть книги посвящена моделям физических процессов в газах и твердых телах, фундаментальным свойствам клеточных автоматов.

Выбор гибкого, расширяемого языка программирования Forth, допускающего возможность создания диалектов, необходимых для решения того или иного класса задач, не случаен. Используемая в нем форма записи выражений - обратная польская запись - позволяет существенно повысить эффективность синтаксического анализа, а стековый характер языка идеально соответствует архитектуре машин клеточных автоматов. При этом значительно упрощается и задача отображения Forth-алгоритмов на СБИС.

Книга несомненно заинтересует многих читателей как превосходное, великолепно иллюстрированное введение в стратегически важную область вычислительной техники и информатики.

Б.В. Баталов

От переводчиков. Уже когда перевод готовился к изданию, в расцвете творческих сил ушел из жизни Борис Васильевич Баталов. Тридцать лет своей жизни Борис Васильевич посвятил становлению микроэлектроники в стране. Его живая мысль, энергия и целеустремленность продолжают жить в делах многочисленных учеников, коллег и друзей. Бориса Васильевича всегда отличали широкая эрудиция, стремление к развитию фундаментальных аспектов микроэлектроники, ясное видение перспектив ее развития. Именно поэтому перевод данной книги осуществлен под его редакцией. Светлая память о Борисе Васильевиче - человеке и ученом, навсегда сохранится в наших сердцах.

БЛАГОДАРНОСТИ

Сочинение этой книги, подобно мирам, которые она описывает, можно было бы продолжать до бесконечности. Мы надеемся, что продолжение этой истории будет написано нашими читателями.

Мы признательны Эду Бартону (E. Barton), Чарльзу Беннетту (C. Bennett), Тому Клони (T. Cloney), Рэю Хиршфельду (R. Hirschfeld), Грводже Грджовчичу (H. Hrgovcic), Марку Смиту (M. Smith), Пабло Тамайо (P. Tamayo), Тао Нгуену (T. Nguyen), Джерарду Вишняку (G. Vichniac) и Дейвиду Зайгу (D. Zaig) за помощь, которую получили в редакционной работе.

Мы хотели бы поблагодарить Харольда Абельсона (H. Abelson), Ричарда Брауэра (R. Brower), Артура Беркса (A. Burks), Николая Кабиббо (N. Cabibbo), Майкла Крейтца (M. Greutz), Доминика д'Юмера (D. d'Humiere), Уриэля Фриша (U. Frisch), Питера Гэкса (P. Gacs), Билла Госпера (B. Gosper), Дейвида Гриффитса (D. Griffeath), Юмана Хартмана (H. Hartman), Бросла Хасслашера (B. Hasslacher), Даниэля Хиллиса (D. Hillis), Джузеппе Якопини (G. Iacopini), Лео Каданова (L. Kadanoff), Рольфа Ландауэра (R. Landauer), Леонида Левина (L.Levin), Майка Левитта (M.Levitt) , Стюарта Нельсона (S. Nelson), Джорджо Паризи (G. Parisi), Ива Помо (Y. Pomeau), Клаудио Ребби (C.Rebbi), Брайана Силвермана (B. Silverman), Джеральда Зуссмана (G. Sussmann) и Стефана Вольфрама (S. Wolfram) за полезные дискуссии и предложения. Чарльз Беннетт (C. Bennett) непосредственно участвовал в написании некоторых материалов, вошедших в текст книги.

Разработка семейства машин клеточных автоматов является ответвлением более теоретических изысканий *Information Mechanics Group* (Группы информационной механики) из Laboratory for Computer Science (Лаборатории информатики) МТИ. Покровительство и практическая поддержка были оказаны директором лаборатории Майклом Дертозосом (M. Dertouzos), Эдвардом Фредкиным (E. Fredkin), который руководил группой до недавнего времени и является автором многих идей, представленных в этой книге, и Джоном Дейчем (J. Deutch).

Эта научно-исследовательская работа была частично поддержана следующими правительственными организациями:

Defense Advanced Research Projects Agency (Grant. No. N00014-83-10-0125), National Science Foundation (Grant No. 8214312-IST), и U.S. Department of Energy (Grant No. DE-AC02-83-ER13082).

Т. Тоффоли *5 ноября 1986 г.*
Н. Марголус

ВВЕДЕНИЕ

В греческой мифологии механизмом Вселенной являлись сами боги. Они самолично тащили солнце по небу, посылали дождь и гром и вкладывали подходящие мысли в человеческие головы. Согласно более новым представлениям, мир создан завершенным, т. е. вместе с механизмом его функционирования; будучи однажды приведен в движение, он продолжает двигаться сам собой. Бог сидит снаружи и наслаждается наблюдением за ним.

Клеточные автоматы являются стилизованными, синтетическими мирами, определенными простыми правилами, подобными правилам настольной игры. Они имеют свой собственный вид материи, которая кружится в своих собственных пространстве и времени. Можно вообразить удивительное разнообразие этих миров. Можно действительно построить их и наблюдать, как они развиваются. Поскольку творцы мы неопытные, вряд ли нам удастся получить интересный мир с первой попытки; как люди, мы можем иметь разные представления о том, что делает мир интересным, или о том, что мы могли бы захотеть сделать с ним. В любом случае, после того как нам покажут мир клеточного автомата, нам захочется сотворить его самим; создав один, мы захотим попытаться создать еще один. После создания нескольких мы сможем создать мир, специально предназначенный для определенной цели, с некоторой уверенностью.

Машина клеточных автоматов является синтезатором миров. Подобно органу, она имеет клавиши и регистры, с помощью которых возможности инструмента можно приводить в действие, комбинировать и перекомпоновывать, а ее цветной экран является окном, через которое можно наблюдать мир, который сейчас "играется".

Итак, эта книга является вводным руководством по гармонии и оркестровке для композиторов миров на клеточном автомате.

Часть I. Обзор

Глава f

КЛЕТОЧНЫЕ АВТОМАТЫ

> То, что сделано однажды, может быть сделано вновь
>
> [Традиционная формула, оправдывающая использование математической индукции]

Синтезировать систему - означает "собрать ее", используя заданный набор понятий, инструментов и материалов. Система может быть абстрактной математической структурой, например дифференциальным уравнением, или конкретным механизмом, скажем телефоном; мы можем интересоваться системой ради нее самой, или как прибором для выполнения конкретной функции, или же как моделью некоторой другой структуры.

В этой книге мы исследуем выразительные возможности (в плане синтеза систем) конкретного набора средств, а именно законов, структур и явлений, поддерживаемых клеточными автоматами, с тем чтобы эти системы стали реально доступными для экспериментирования посредством использования машины клеточных автоматов с адекватными характеристиками.

Данная глава является введением в клеточные автоматы и включает краткие исторические замечания и ссылки.

1.1. Основные понятия

Клеточные автоматы являются дискретными динамическими системами, поведение которых полностью определяется в терминах локальных зависимостей, в значительной степени так же обстоит дело для большого класса непрерывных динамических систем, определенных уравнениями в частных производных. В этом смысле *клеточные автоматы в информатике являются аналогом физического понятия "поля"*.

Как отмечено во введении, клеточный автомат может мыслиться как стилизованный мир. Пространство представлено равномерной сеткой, каждая ячейка которой, или *клетка,* содержит несколько битов данных; время идет вперед дискретными шагами, а законы мира выражаются единственным набором правил, скажем небольшой справочной таблицей, по которой любая клетка на каждом шаге вычисляет свое новое со-

стояние по состояниям ее близких соседей. Таким образом, законы системы являются *локальными и повсюду одинаковыми.*[1]

Если задан подходящий набор правил (рецепт), то такой простой операционный механизм достаточен для поддержания целой иерархии структур и явлений. Клеточные автоматы дают полезные модели для многих исследований в естественных и вычислительных науках и комбинаторной математике; они, в частности, представляют естественный путь изучения эволюции больших физических систем. Клеточные автоматы к тому же образуют общую парадигму параллельных вычислений, подобно тому как это делают машины Тьюринга для последовательных вычислений.

1.2. Мультипликация вручную

Прежде чем поручать машине управлять миром на клеточном автомате, необходимо иметь ясное представление о том, как то же задание могло бы быть выполнено вручную.

Возьмите стопку миллиметровки или другой разграфленной на клетки бумаги хорошего качества, в которой сетка напечатана в точности на том же месте на каждом листе. Начиная с последнего листа и проходя стопку в обратном направлении, вы должны нарисовать серию кадров, по одному на каждом листе, которые составят краткую мультипликационную последовательность.

Откройте стопку на последнем листе и нарисуйте простую картинку, заполняя несколько клеток сетки рядом с центром; это будет первый кадр вашего мультипликационного фильма. Переверните затем страницу, наложив предпоследний лист на последний; рисунок первого кадра будет просвечивать сквозь бумагу. Нарисуйте очередной кадр фильма на новом листе по следующему рецепту INKSPOT:

> Выберите клетку и посмотрите на область 3x3, центром которой она является, - ее "окрестность". Если вы увидите в этой области (на листе, находящемся прямо под этим) в точности три черные клетки, пометьте слегка вашу клетку карандашом; также пометьте эту клетку, если она находится непосредственно над черной клеткой.

[1] "Локальный" означает, что для того, чтобы узнать, что произойдет здесь мгновение спустя, достаточно посмотреть на состояние ближайшего окружения: никакое дальнодействие не допускается. "Одинаковость" означает, что законы везде одни и те же: я могу отличить одно место от другого только по форме ландшафта, а не по какой-то разнице в законах.

Сделайте то же для каждой клетки решетки. По окончании закрасьте чернилами все отмеченные клетки.

Постройте третий кадр из второго, обращаясь к следующему листу и применяя тот же способ; затем постройте четвертый кадр из третьего и так далее, пока не будут использованы все листы.

Теперь вы можете взглянуть на ваш блокнот. Сложите стопку, захватив ее край между большим и указательным пальцами, и пусть листы падают один за другим в быстрой последовательности. То, что вы увидите, является чернильным пятном, которое проступает нерегулярно - как на грязной материи - порождая мысы и заливы, оставляя незначительное число маленьких областей нетронутыми и время от времени выбрасывая прямые нити (рис. 1.1).

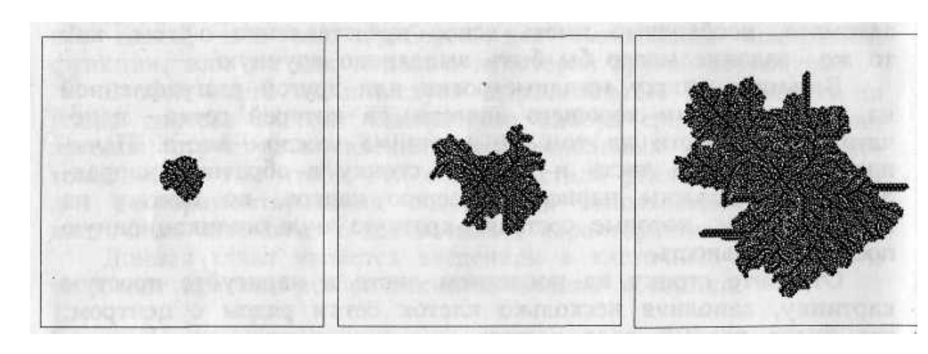


Рис. 1.1. Несколько кадров типичной последовательности INKSPOT. Масштаб таков, что отдельные клетки едва различимы.

Это в сущности все, что нужно для клеточного автомата. Измените начальные условия, и вы получите несколько иную историю. Измените рецепт, и вы получите новое множество "динамических законов - новый мир. Вы можете использовать сетку иной формы (скажем, гексагональную), или, может быть, трехмерную. Рецепт может относиться к окрестности с другой формой и размером, чем окрестность 3x3, и может включать чернила более чем одного цвета; однако как число соседей, так и число цветов (т. е. число возможных *состояний клетки)* должны быть конечными, потому что мы хотим, чтобы обновление состояния клетки требовало конечного числа операций.

Предположим, что наш лист содержит 100.000 клеток (это несколько грубее, чем разрешающая способность телевизионного кадра). Если рецепт не слишком сложный, то потребуется один день художника-мультипликатора, чтобы нарисовать новый кадр; задание предельно простое, как и "рисование по цифрам", но долгое, утомительное и подверженное ошибкам.

Персональный компьютер сделает эту работу за несколько секунд, генерируя новые кадры с частотой показа слайдов. Если мы хотим увидеть настоящее движение, необходимо двигаться, вероятно, в тысячу раз быстрее: универсальный суперкомпьютер может это делать, но очень дорого и с совершенно очевидным разбазариванием его ресурсов. Желателен более эффективный подход.

1.3. Машины клеточных автоматов

Давайте прямо признаем, что общность и гибкость клеточно-автоматного подхода к синтезу систем достигаются не бесплатно. Вместо небольшого число переменных, взаимодействие которых может быть задано произвольным образом, клеточный автомат использует много переменных (одна на клетку), но требует, чтобы они взаимодействовали только локально и единообразно.

Чтобы синтезировать структуры значительной сложности, необходимо использовать большое количество клеток, а для того чтобы эти структуры взаимодействовали друг с другом и существенно эволюционировали, необходимо позволить автомату работать на протяжении большого количества шагов. Для элементарных научных проблем удовлетворительная экспериментальная работа может потребовать вычисления миллиардов событий *(событием* является обновление одной клетки); для более сложных приложений может быть желательным значение в тысячу или миллион раз больше (т. е. 10^{12}-10^{15} событий); в действительности пределы устанавливаются тем, сколько мы *можем* выполнить, а не тем, сколько хотим.

В связи с этим обычные компьютеры здесь мало пригодны. Моделирование события в клеточном автомате может потребовать около тридцати машинных операций, содержащих каждая несколько машинных циклов, скажем 10 мс на быстродействующем компьютере. Для того чтобы вычислить 10^{13} событий, при таком подходе потребовалось бы несколько лет!

С другой стороны, структура клеточного автомата идеально пригодна для реализации на ЭВМ, обладающей высокой степенью параллелизма, локальными и единообразными взаимо-

связями[1]; подходящая архитектура позволяет при моделировании клеточных автоматов за сравнимую цену достигнуть эффективности по меньшей мере на несколько порядков выше, чем для обычного компьютера.

И в самом деле, машины клеточных автоматов, имеющие размеры, скорость и гибкость, подходящие для общего экспериментирования, и умеренную стоимость, стали в последнее время доступны широким научным кругам (см. гл. 2). Эти машины представляют собой лабораторные установки, в которых идеи, приведенные в этой книге, могут быть испытаны в конкретной форме и применены к синтезу огромного многообразия систем.

1.4. Исторические замечания и литература

Клеточные автоматы изобретались много раз под разными названиями, и несколько отличающиеся друг от друга понятия употреблялись под одним и тем же названием. В чистой математике их можно обнаружить как один из разделов топологической динамики, в электротехнике они иногда называются итеративными массивами, а студенты младших курсов могут знать их как вид игры на домашнем компьютере. Их использовали и злоупотребляли ими не только междисциплинарные ученые, но и междисциплинарные болтуны. Они стали темой или поводом бесчисленных диссертаций. О них много говорилось и писалось, но до последнего времени никто в действительности не *видел* большинство из них.

Поскольку эти исторические замечания предназначены для того, чтобы помочь читателю, а не сбить его с толку, мы коснемся только тех тем, которые считаем непосредственно относящимися к целям этой книги.

В обычных моделях вычислений, таких как машина Тьюринга, различают структурную часть компьютера, которая фиксирована, и данные, которыми компьютер оперирует - они являются переменными. Компьютер не может оперировать своей собственной "материальной частью"; он не может себя расширять или модифицировать, строить другие компьютеры.

Клеточные автоматы ввел в конце сороковых годов Дж. фон Нейман, следуя идее С. Улама [64], для того чтобы обеспечить более реалистические модели поведения сложных, пространственно протяженных систем [68]; в клеточном авто-

[1] Термин "не-фон-неймановская архитектура" часто используется для различения параллельных компьютеров этого вида и более привычных последовательных компьютеров. Необходимо, однако, заметить, что теория клеточных автоматов была введена самим фон Нейманом примерно в то же время, когда он работал над конструированием универсальных электронных компьютеров.

мате и объекты, которые могут быть интерпретированы как пассивные данные, и объекты, которые могут быть интерпретированы как вычислительные устройства, собираются из одного типа структурных элементов и подчиняются одним и тем же "мелкозернистым" законам; вычисление и конструирование являются просто двумя возможными типами активности.

Хотя фон Нейман был ведущим физиком в такой же степени, как и математиком, точные физические рассуждения отсутствуют в его работе по клеточным автоматам; его больше интересовало редукционистское объяснение определенных аспектов биологии. Действительно, механизмы, которые он предложил для получения самовоспроизводящихся структур на клеточном автомате, сильно напоминают открытые в следующем десятилетии механизмы, которые на самом деле наблюдаемы в биологических системах.

В конце войны, когда фон Нейман создавал один из первых электронных компьютеров, немецкий инженер К. Цусе прятался от нацистов в Австрии; там, в уединении на вершине горы, у него возникли наброски многих идей параллельной обработки, включая языки программирования высокого уровня и "вычисляющие пространства" [76] - т. е. клеточные автоматы. К. Цусе особенно интересовался численными моделями в механике, и физические мотивы играли основную роль в его работе. К несчастью, исторические обстоятельства воспрепятствовали более широкой известности его работ в то время.

Работа фон Неймана по самовоспроизводящимся автоматам была завершена и описана А. Берксом [68], который сохранил активный интерес к этой области на протяжении нескольких последующих лет. Его "Очерки по клеточным автоматам" [10] являются хорошим введением в вопросы о клеточных автоматах, которые ставились в годы формирования вычислительных наук. В той же среде, т. е. в группе компьютерной логики Университета шт. Мичиган, Дж. Голланд приступил к использованию клеточных автоматов в задачах адаптации и оптимизации [27]; был разработан программный имитатор универсальных клеточных автоматов общего назначения. Месяцы работы с этим имитатором (см. *[55])* убедили одного из авторов (Тоффоли) в необходимости более непосредственной и эффективной аппаратной реализации - *машины* клеточных автоматов.

Тем временем профессиональные математики обратили внимание на итерационные преобразования, действующие на пространственно распределенные структуры с дискретным набором состояний [25], - опять-таки клеточные автоматы! Недостаточность общения и единой терминологии вели к значительному дублированию работ. Важные характерные особенности клеточных автоматов, доказанные Ричардсоном [48] на

двадцати страницах в континуальной постановке в топологии канторовских множеств, могли бы в действительности быть записаны в двух строках в виде следствия к предшествующей работе Хедлунда [25]. Аналогично, прямой "силовой" поиск сюръективных клеточных автоматов, доложенный Пэттом в 1971 г. (см. [2]), был проведен в более широком масштабе Хедлундом и др. [24] уже в 1963 г.!

Важные теоретические вопросы обратимости и вычислимости, затронутые Муром и Майхиллом [10], были наряду с другими исследованы А. Смитом [52], С. Аморозо [2] и В. Аладьевым [1]; этот подход был продолжен и ныне процветающей японской школой (см. [40] и литературу к этой работе).

Игра Джона Конвея "жизнь", представленная широкой общественности ведущим рубрику математических игр и развлечений в журнале "Сайентифик Американ" М. Гарднером [20], некоторое время пользовалась популярностью, близкой к культу, и сделала выражение "клеточные автоматы" частью бытового жаргона целого поколения молодых ученых.

Мы интересуемся клеточными автоматами прежде всего как *автономными* системами, т. е. как мирами, замкнутыми в себе, а не как *трансдьюсерами* (системы, которые порождают постоянный выходной поток информации в ответ на постоянный входной поток). По этой причине мы совсем не будем касаться обширной литературы, посвященной итерационно-цепным массивам в контексте арифметических вычислений, обработки изображений и распознавания образов. В качестве введения в эти области и руководства к машинам, разработанным для этих более специализированных приложений, может быть использована книга Престона и Даффа "Современные клеточные автоматы" [46].

Вопрос о том, могут ли клеточные автоматы моделировать непосредственно законы физики, а не только общие феноменологические аспекты нашего мира[1], был вновь поставлен Э. Фредкином, который проявлял активность и в более традиционных областях исследований клеточных автоматов (см. [3]), и Т. Тоффоли *[55].* Основной целью настоящего исследования является формулировка компьютероподобных моделей в физике, *сохраняющих информацию,* а значит и одно из наиболее фундаментальных свойств микроскопической физики, а именно *обратимость* [17, 58, 35].

Модели, которые явным образом сводят макроскопические явления к точно определенным микроскопическим процессам, представляют наибольший методологический интерес [13], по-

[1] Такие как общение, вычисление и конструирование; рост, воспроизведение, конкуренция, эволюция и т.д.

тому что они обладают огромной убедительностью и ясностью (см. гл. 13). Но для того, чтобы они вообще могли что-то нам сказать, в общем случае нет иного выхода, кроме непосредственной реализации предписаний этих моделей, на деле преодолевающей пропасть между микроскопическим и макроскопическим масштабами: имитаторы клеточных автоматов, способные обновлять состояния миллионов клеток за предельно короткое время, становятся незаменимыми инструментами. В этом состоит одна из задач, к которой обратилась наша группа информационной механики в Лаборатории информатики МТИ в связи с конструированием высококачественных машин клеточных автоматов [59, 60, 36].

Этот подход был использован для того, чтобы обеспечить предельно простые модели обычных дифференциальных уравнений физики, таких как уравнение теплопроводности, волновое уравнение [61] и уравнение Навье-Стокса [23, 18], которые могут мыслиться как предельные случаи исключительно простых процессов комбинаторной динамики. В частности, клеточные автоматы были созданы для того, чтобы дать точные модели динамики жидкостей, которые не только будят мысль, но и конкурентоспособны, по крайней мере в некоторых обстоятельствах, с точки зрения их вычислительной эффективности.

Бурно развивающийся раздел теории динамических систем изучает возникновение хорошо описанных коллективных явлений - упорядочение, турбулентность, хаос, нарушение симметрии, фрактальность и др. в системах, состоящих из большого числа частиц, взаимодействующих друг с другом нелинейно; цели исследований и их математический аппарат здесь больше похожи на присущие макроскопической физике и материаловедению. Клеточные автоматы обеспечивают богатую и непрерывно растущую коллекцию типичных моделей, в которых эти явления могут быть изучены относительно легко [66, 15, 5]. Систематическое использование клеточных автоматов в этом контексте энергично проводилось С. Вольфрамом [70-73, 43]; его сборник статей по теории и применениям клеточных автоматов [74] содержит обширную библиографию.

Итак, клеточные автоматы, по-видимому, нашли устойчивое (и все более важное) применение в качестве концептуальных и практических моделей пространственно-распределенных динамических систем, для которых физические системы являются первыми и наиболее важными прототипами.

Глава 2

СРЕДА САМ

> Итак, засучив рукава и призвав все свое мастерство, Трурл построил королю совершенно новое королевство...
>
> "Я правильно тебя понял? - спросил Клапауциус. - Ты подарил этому жестокому деспоту ... целую цивилизацию для того, чтобы править и властвовать вечно? Трурл, как ты смог это сделать?!"
>
> "Ты, должно быть, шутишь! - воскликнул Трурл. - В действительности все королевство помещается в ящике размером три на два с половиной фута... это только модель..."
>
> "Модель чего?"
>
> [Станислав Лем]

Обычным предварительным условием для изучения курса гармонии является "знакомство с фортепьяно". Конечно, гармонию можно изучить и множеством других способов - например, на органе, гитаре или с помощью хора. Однако стандартная "среда" помогает сосредоточить внимание на вопросах более принципиального характера: это уже дело самого студента перенести в другие обстоятельства опыт, полученный при овладении возможностями этой среды.

В этой книге стандартная среда моделирования, которая будет постепенно вводиться, начиная с настоящей главы, представлена конкретной имеющейся в продаже машиной клеточных автоматов, а именно САМ-6. Основная причина этого выбора состоит в том, что аппаратные средства и программное обеспечение этой машины доступны в настоящее время широкому кругу пользователей.

2.1. Машина САМ-6

САМ-6 является машиной клеточных автоматов, предназначенной для того, чтобы служить лабораторией экспериментатора, средством сообщения результатов и средой для интерактивной демонстрации в режиме реального времени.

Машина была первоначально разработана в Лаборатории информатики Массачусетсского технологического института (МТИ). В настоящее время она производится фирмой SYSTEMS CONCEPTS (Сан-Франциско, СА), которая ее распространяет с

ясно сформулированной целью: после удовлетворения внутренних нужд МТИ сделать дальнейшую продукцию производственного конвейера доступной широким научным кругам по настолько низкой, насколько это возможно, цене.[1]

Физически CAM-6 состоит из модуля, который вставляется в один разъем IBM-PC (XT, AT или совместимых с ними моделей), и управляющего программного обеспечения, работающего в среде PC-DOS2. В то время как этот легко доступный головной компьютер обеспечивает размещение, экранирование, электропитание, дисковую память, монитор и стандартную операционную среду, вся действительная работа по моделированию клеточных автоматов с очень высокой скоростью совершается самим модулем с быстродействием, сравнимым (для этого частного приложения) с быстродействием CRAY-1.

Управляющее программное обеспечение для CAM-6 написано на FORTH и работает на IBM-PC с памятью 256 K. Это программное обеспечение дополнено рядом готовых приложений и демонстрационных примеров и включает полный аннотированный список источников.

Сама система FORTH, полученная из модели F83 Лаксена и Перри,[2] является общедоступной и сопровождается полной входной информацией, снабженной примечаниями.

В остальной части этой книги мы будем называть CAM-6 просто CAM.

2.2. Основные аппаратные средства

Здесь мы дадим краткий обзор тех основных аппаратных средств CAM, которые видны пользователю и составляют "программистскую модель" машины. Более полное обсуждение этих и других средств откладывается до следующих глав.

2.2.1. Память: плоскость битов

В примере мультипликации числами из разд. 1.2 давайте зафиксируем наше внимание на конкретной клетке и проследим ее историю от кадра к кадру. В каждый момент данная клетка будет или белой, или черной. Таким образом, каждая клетка может мыслиться как переменная с двумя возможными

[1] Машины семейства CAM используются уже несколько лет. Более ранняя версия, CAM-5, была описана в *Physica D* [59], а популярные заметки, связанные с ней, появились в *Scientific American* [22], *High Technology* [63] и *Discover* [47]. CAM-6 находится на постоянной выставке в музее компьютеров в Бостоне.

[2] В действительности эта работа выполнена значительно большим числом разработчиков.

состояниями, или *бит,* а вся решетка как двумерный массив битов, или *плоскость битов.* Два возможных значения бита будет удобно называть о и 1, а не белое и черное.

Во многих приложениях необходимы клетки с богатым набором состояний. Предположим, что нам нужны четыре состояния. Вместо только о и 1 мы можем использовать символы О, 1, 2 и 3 как возможные состояния клетки. Другой способ - разделить клетку на две подклетки, содержащие каждая по одному биту, и записать четыре состояния как оо, 01, ю, 11; в этом случае полезно мысленно представить себе эти два бита размещенными один над другим, а не размещенными бок о бок. Тогда весь массив можно мысленно представить как набор двух *плоскостей битов,* расположенных одна над другой.

В САМ доступно до четырех плоскостей битов для кодирования состояния клетки, и, следовательно, клетка может иметь до 16 состояний. (Есть, однако, определенные ограничения на совместное использование четырех плоскостей битов.)

2.2.2. Дисплей: цветовая карта

Трактовка состояния клетки как стопки четырех битов удобна с точки зрения программирования. Для целей наглядной демонстрации лучше представить каждую клетку цветной точкой на экране монитора, или *пикселем.* Цвет, конечно, будет соответствовать состоянию клетки согласно определенному соглашению: таблица, которая определяет, какой цвет обозначает каждое из 16 возможных состояний клетки, называется *цветовой картой.* В САМ пользователь может определить содержание цветовой карты так, чтобы оно удовлетворяло требованиям каждого конкретного эксперимента.

2.2.3. Динамика: таблицы правил

В машине клеточных автоматов текущий кадр, представленный содержимым всех плоскостей битов, во время цикла обновления заменяется новым согласно определенному рецепту. Результатом является один *шаг* эволюции конкретного клеточного автомата, а этот рецепт называется *правилом* этого клеточного автомата.

В САМ пользователю разрешается определять правила весьма произвольным образом, используя конструкции из языков высокого уровня. Внутри, однако, это определение в конечном счете преобразуется в *таблицу* правил, в которой в явном виде перечислено, каким будет новое состояние клетки для всякой возможной комбинации состояний ее соседей. Поскольку каждая клетка состоит из четырех битов, будет удобно представить таблицу правил, состоящей из четырех подтаблиц, или

компонент, по одной на каждую плоскость битов. В этом смысле мы будем считать, что выражение "правило для плоскости 0" означает "те компоненты (общего правила), которые определяют новое состояние плоскости битов О".

2.2.4. Геометрия в малом: окрестность

При написании правила клеточного автомата мы определяем, как на каждую клетку повлияют некоторые соседние клетки. Точнее, мы определяем, как на каждый из четырех битов, составляющих клетку, повлияют несколько соседних битов; некоторые из них могут располагаться на той же самой плоскости битов, а некоторые на остальных трех плоскостях. Насколько далеко может распространяться это влияние?

Бит называется *соседом* другого, если у него есть возможность прямо воздействовать на него согласно правилу за один шаг. В принципе правило клеточного автомата могло бы использовать любое число соседей. Однако соображения эффективности диктуют практический предел числа и длины прямых связей соседей. Аппаратные средства CAM обеспечивают специальные комбинации связей соседей, или *окрестности,* которые были выбраны в соответствии с критериями общей пользы и гибкости. Выбор окрестности обсуждается в гл. 7.

2.2.5. Геометрия в большом: обертывание

Возвращаясь к примеру мультипликации числами, задумаемся, как применить правило INKSPOT, когда клетка лежит на краю листа, так что некоторых из ее соседей недостает в кадре? Конечно, мы могли бы в явном виде задать рецепты для этого особого случая, но было бы лучше вовсе избежать необходимости вводить "особые случаи".

В CAM эта проблема решена с помощью очевидного приема *обертывания,* т. е. правый край листа ведет себя так, как если бы он был "приклеен" к левому краю (и аналогично для верхнего и нижнего краев). Как и во многих видеоиграх, любой объект, который пытается уйти за край экрана, вернется на экран в той же позиции у противоположного края.[1] Кадр можно себе представить нарисованным на поверхности автомобильной камеры: сколь угодно долгий путь не позволит найти "край света"/ Дополнительные варианты обертывания упомянуты в разд. В.2.

[1] Несколько модулей CAM могут быть объединены так, чтобы получить больший клеточный автомат. В этом случае "склеивание" краев соответствующим образом модифицируется, так что обертывание правильно применяется и к большему листу.

[2] Более строго, топология соответствует тору.

2.3. Программное обеспечение: CAM Forth

Пользователю CAM обеспечивается доступ к программному обеспечению на различных уровнях, как разъясняется в документации, которая сопровождает машину. Для наиболее простых приложений, однако, программное обеспечение прозрачно для пользователя, и CAM можно считать весьма похожей на электробытовой прибор, в котором несколько строк кнопок непосредственно управляют выбором режимов работы в интерактивном режиме.

Однако, если вы хотите уметь добавлять новые элементы к своему набору миров, вам для описания правил клеточного автомата потребуется язык, удобный пользователю и понятный машине. Этот процесс создания правил формально вводится в гл. 4, а полностью разрабатывается в последующих главах.

В этой книге будет использоваться язык CAM *Forth*. Forth является расширяемым языком программирования, особенно подходящим для интерактивных задач. Этот язык был расширен так, чтобы он содержал множество слов и конструкций, полезных для поддержания диалога с CAM (в частности, для определения правил клеточного автомата и для конструирования, документирования и выполнения экспериментов).

Forth был принят в качестве стандартного языка Международным астрономическим союзом с целью способствовать обмену процедурами наведения телескопов и управления вспомогательным оборудованием. Здесь мы будем пользоваться им для настройки нового вида "скопа", позволяющего увидеть множество синтетических миров.

Для определения правил клеточного автомата на самом деле нужно очень маленькое подмножество Forth. С другой стороны, структура программ на Forth необычна, и даже людям, которые уже знакомы с более традиционными языками программирования (такими как Паскаль или Бейсик), может быть полезно минимальное введение. Приложение A представляет краткое введение, охватывающее наиболее важные понятия; необходимо взглянуть на него до работы. Разделы A. 12 и A. 14 содержат, в частности, информацию, которая характерна для этой версии Forth. Часть вспомогательного материала будет напоминаться и расширяться по ходу изложения; конструкции, которые более тесно связаны с работой CAM, будут вводиться там, где это необходимо.

Для более полного знакомства с языком читатель может обратиться к книге Л. Броди [8], которая является превосходным введением в предмет.

Глава 3

ЖИВАЯ ДЕМОНСТРАЦИЯ

> И вот я совершенно ясно увидел, что это были очень маленькие угри, или змеи, лежащие свернувшись в комок и извивающиеся; точно также как если бы вы увидели невооруженным глазом целую бочку воды и очень маленьких угрей, извивающихся один среди других: при этом вся вода кажется кишащей этими анималькулями.
>
> [Левенгук]

Прочитайте эту краткую главу один раз, не слишком вникая в детали. Представьте, что ваш товарищ сидит у пульта управления CAM машиной, демонстрируя вам краткую последовательность экспериментов и попутно объясняя, что он делает и почему. Вы можете отложить вопрос, чтобы не отрывать его, и во многих случаях ответ на него будет дан на экране. В конце концов вы можете потребовать пройти несколько пунктов более детально.

К концу занятия вы не рассчитываете "знать все". Однако вы хотите почувствовать, что есть определенная связь между поставленными целями, выполненными действиями и увиденными результатами, и что при лучшем знакомстве с имеющимися командами и средствами вы могли бы выполнить некоторые эксперименты самостоятельно.

В качестве объекта для этой демонстрации мы выбрали игру LIFE ("жизнь"). Это не единственное, и даже не наиболее интересное правило клеточного автомата, а просто наиболее широко известное, и появилось оно здесь скорее по историческим причинам, а не из-за какой-то прямой связи с темой книги.

3.1. Хгра "жизнь"

В 1970 году математик Джон Конвей ввел очаровательный клеточный автомат, который привлек внимание любителей и профессионалов во всем мире [20]. "Жизнь" можно считать игрой, описывающей популяцию стилизованных организмов, развивающуюся во времени под действием противоборствующих тенденций размножения и вымирания.

Индивидуум этой популяции представлен клеткой в состоянии 1, в то время как о представляет пустое пространство;

для краткости можно соответственно говорить о "живых" и "мертвых" клетках. На каждом шаге любая клетка реагирует на состояние ее непосредственного окружения, состоящего из восьми ближайших соседей, согласно следующим правилам:

Смерть: Живая клетка остается живой, только когда она окружена двумя или тремя живыми соседями; в противном случае она будет чувствовать или "перенаселенность", или "одиночество" и умрет.

Рождение: Мертвая клетка обретет жизнь, если будет окружена в точности тремя живыми соседями. Таким образом, рождение вызывается встречей трех "родителей".

В CAM внутренний механизм "знает", что ко всем клеткам должно быть применено одно и то же правило, поэтому достаточно выразить правило для "общей" клетки, именуемой CENTER (центр внимания). Восемь соседей этой клетки называются NORTH, SOUTH, WEST, EAST, N.WEST, N.EAST, S.WEST и S.EAST; смысл этих имен ясен. Для того чтобы написать правило для LIFE в CAM Forth, мы сначала определим слово 8SUM, которое будет подсчитывать количество живых соседей, а затем использовать это количество как адрес для нахождения элемента данных в таблице, который соответствует этому конкретному числу; данные сами определяют новое состояние клетки. В нашей программе имеются две таблицы, одна для случая "мертвой" центральной клетки, а другая для "живой".

```
                              : 8SUM (•-- количество)
          NORTH SOUTH WEST EAST      \ количество варьирует
     N.WEST N.EAST S.WEST S.EAST     \  от 0 до 8
                    + + + + + + + ;
                              : LIFE
                       CENTER 0= IF
8SUM { 0 0 0 1 0 0 0 0 0 } ELSE      \ таблица мертвых клеток
8SUM { 0 0 1 1 0 0 0 0 0 } THEN      \ таблица живых клеток
                           >PLN0;    \ правило для плоскости битов 0
```

Пока что вам не надо понимать приведенную выше программу на Forth (хотя сейчас было бы полезно заглянуть в руководство в приложении А). Во всяком случае, просмотр определения 8SUM должен показать правдоподобность того, что это слово языка Forth складывает восемь величин (здесь восемь слагаемых и семь знаков +). Выражение в скобках (— количество) есть просто комментарий и напоминает нам, что результатом 8SUM является желаемое количество соседей. Аналогично,

правдоподобно и то, что слово LIFE языка Forth может так или иначе использовать значения 8SUM, для того чтобы выбрать конкретный элемент в списке девяти объектов, и что значение CENTER может определять, какой из двух данных списков действительно используется. Наконец, слово >PLNO сообщает программе, что объект, определенный этим способом, является новым состоянием бита в плоскости битов 0 машины CAM.

Давайте запустим игру "жизнь" на экране с "первичным бульоном", в котором нули и единицы распределены случайно и в равных количествах (рис. 3.1а). После нескольких дюжин шагов бешенной активности популяция поредеет (рис. 3.1Ь); чуть позже большая часть экрана будет неподвижной, за исключением нескольких мест, где жизнь тлеет и иногда вспыхивает внезапными всплесками активности, что напоминает бои местного значения (рис. 3.2а). В конце концов вся активность может угаснуть, за исключением нескольких изолированных "мигалок" или других образований, развивающихся циклически с коротким периодом (рис. 3.2Ь).

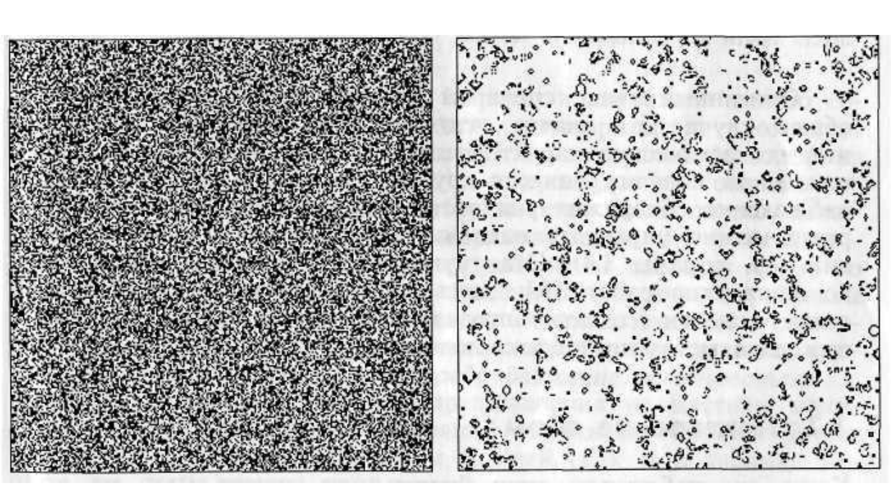


Рис. 3.1. Кадры игры LIFE: (а) исходная случайная конфигурация; (Ь) несколько дюжин шагов спустя "первичный бульон" поредел.

Для того чтобы статическое изображение позволило почувствовать динамическое поведение, кадры на рис. 3.2 получены усреднением по времени (сняты с продолжительной выдержкой).

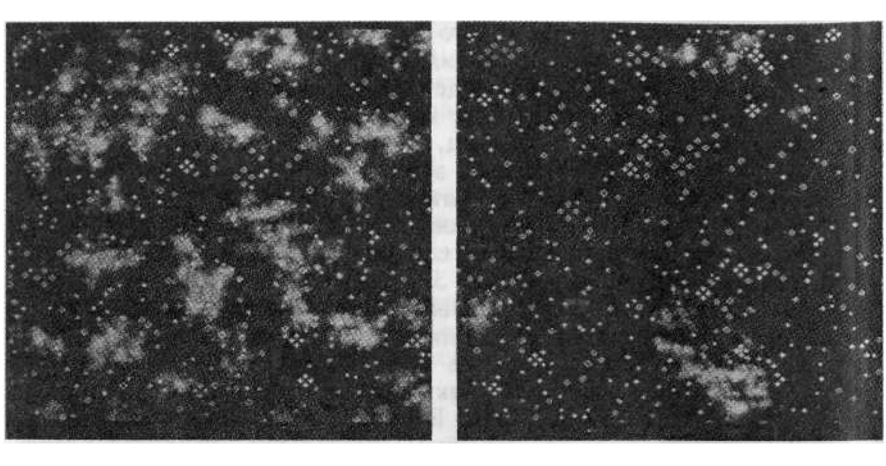


Рис. 3.2. (а) Несколько сотен шагов спустя активность концентрируется в нескольких тлеющих областях, с редкими вспышками. (Ь) Несколько тысяч шагов спустя почти вся деятельность затухла.

Описанный выше сценарий типичен для игры "жизнь"; в общем случае это значит, что поведение, имеющее аналогичные стохастические характеристики возникает из почти любой начальной конфигурации; с другой стороны, более пристальное наблюдение обнаружит ряд четко выраженных эффектов, которыми можно управлять надлежащим выбором начальных условий (ср. с разд. 3.4). Эта группа явлений подробно описана во множестве работ [6]; здесь мы используем игру "жизнь" лишь с целью создать определенный контекст для иллюстрации ряда методологических положений.

3.2. Повторение эхом

Когда мы наблюдаем игру "жизнь" на экране CAM, на всей его площади одновременно происходит множество событий, и мы можем пожелать замедлить или даже остановить на время моделирование, чтобы рассмотреть некоторые детали. Однако как раз тогда, когда темп достаточно медленный, чтобы мы могли не спеша рассмотреть отдельный кадр, ощущение движения теряется; события, которые на полной скорости выглядели совершенно отличными друг от друга благодаря их динамичности, теперь выглядят разочаровывающе одинаковыми, и нелегко запомнить, какая клетка участвует в этих событиях, а какая принадлежит неподвижному фону. Перефразируя Зенона, при свете электронной вспышки трудно отличить летящую пу-

лю от кусочка жевательной резинки. То, что нам нужно, это способ окрашивания в цвета "скорости" элементов черно-белой открытки, представляющей текущее положение.

Клетки игры "жизнь" состоят из одного бита каждая. Таким образом, в ней используется единственная плоскость битов CAM, и ее можно изобразить на черно-белом экране. Но CAM имеет несколько плоскостей битов и может управлять цветным монитором. Когда мы конструируем новое состояние клетки в плоскости 0, то можем вместо отбрасывания предшествующего состояния, временно сохранить его на второй плоскости битов, скажем плоскости 1 (здесь можно представить плоскость 1 как эхо прошлого). Таким способом мы можем собрать двухбитовое значение из нового и старого состояния и можем дать команду цветовой карте CAM изобразить различными цветами каждую из четырех комбинаций 00, 01, 10, 11.

Правило, которое делает плоскость 1 эхом плоскости 0 с задержкой на один шаг, это просто

```
: ECHO
CENTER >PLN1;
```

Заметим, что несмотря на то, что это правило для нового состояния плоскости 1, CENTER по-прежнему означает центрального соседа из плоскости 0; центральный сосед из плоскости 1, именуемый CENTER', совершенно не "ощущается" этим правилом, так как эхо отбрасывается после одного шага.

Если мы перезапустим эксперимент разд. 3.1 с включенным ECHO, используя назначения цветов из таблицы (3.1), то наша игра станет значительно красочнее. Статические объекты выглядят красными, а движущиеся объекты выглядят светло-зелеными у переднего края и светло-голубыми у заднего; *цвета сохранятся даже тогда, когда движение остановится.*

Цвета, используемые для представления на цветном мониторе четырех возможных состояний клетки из четырех битов, - красный, зеленый, голубой и черный, как и в таблице ниже; эта схема цветов применяется и во вкладных цветных иллюстрациях, которые были взяты прямо с экрана монитора. Конечно, для черно-белых рисунков необходима другая схема; она указана в той же таблице.[1] Действительные цвета, во избежание путаницы, будут упоминаться в оставшейся части книги только при ссылках на цветные иллюстрации; для остальных иллюстраций мы должны представить себя пользователями, имеющими "бумажный монитор", способный изображать черный цвет, белый и два промежуточных оттенка серого. Ри-

[1] На бумаге большие области фона лучше оставить белыми (без чернил), а не черными, как на мониторе.

сунки, изображающие единственную плоскость битов, такие как рис. 3.1, будут, конечно, использовать черный цвет для 1 и белый цвет для о.

СТАТУС КЛЕТКИ (LIFE с ECHO)	ПЛОСКОСТЬ О	1	ПЕЧАТНАЯ СТРАНИЦА	ЦВЕТНОЙ МОНИТОР
оставшаяся живой	1	1	Черная	Красная
только что рожденная	1	0	Темносерый	Зеленый
только что умершая	0	1	Светлосерый	Синий
оставшаяся мертвой	0	0	Белый	Черный

3.3. Трассировка

Свойство ECHO обогащает картинку небольшим количеством кратковременной памяти. В принципе эхо можно продлить, используя больше вспомогательных плоскостей битов для хранения информации о нескольких последних шагах в конвейерном режиме, но в любом случае существуют пределы количества исторической информации, которая может быть эффективно изображена на экране за счет кодирования цветом или интенсивностью.

Во многих ситуациях будет эффективным более простой способ *трассировки,* в частности, когда число развивающихся на однородном фоне объектов ограничено - как в игре "жизнь". Мы допустим, что каждая живая клетка в плоскости О оставляет след своего присутствия в плоскости 1, как в случае с ECHO, но на этот раз след будет *постоянным* (как если бы мы ввели в электронно-лучевую трубку монитора слой люминофора с неограниченным временем послесвечения). Режим TRACE включается установлением для плоскости 1 правила

```
                                        : TRACE
                  CENTER CENTER' OR >PLN1;
```

При том же кодировании цвета, что и в ECHO (интерпретация, конечно, слегка отлична), движущиеся объекты оставят за собой светло-серый след, а отлив оставит берег заштрихованным светло-серым цветом вплоть до верхней отметки прилива. Когда лист трассировки станет слишком закрашенным, мы начнем новый, сделав по крайней мере один шаг в режиме ECHO (это стирает долговременную историю) и возобновляя TRACE всякий раз, когда это желательно.

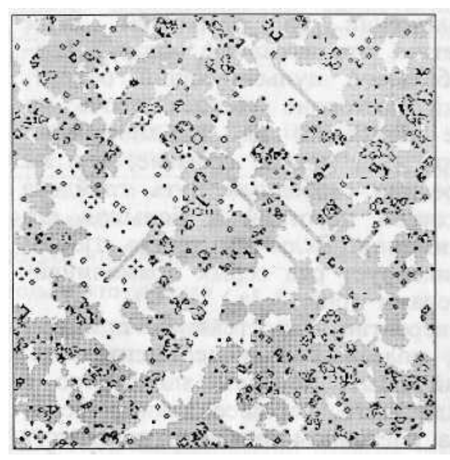
Давайте продолжим наблюдение за развитием игры "жизнь" в общем случае. Отметим, что активные области время от времени извергают небольшой пульсирующий объект, именуемый *глайдером,* который постоянно убегает прочь по диагональному пути, пока не наткнется на что-либо еще. Спонтанное порождение глайдера - не редкое событие; тем не менее в обычных условиях средний свободный путь глайдера (от порождения до распада) совсем короткий и большая его часть ускользнет от нашего внимания. Включим теперь TRACE; за несколько минут экран зафиксирует ряд серых прямолинейных следов, оставленных глайдерами (рис. 3.3а.)

ECHO и TRACE - это, вероятно, простейшие примеры средств *усиления изображений,* полезных в экспериментах на клеточном автомате. Некоторый опыт, хорошее знакомство с доступными ресурсами и, прежде всего, хорошее знание феноменологии конкретного мира на клеточном автомате подскажут целый ряд средств наблюдения, обладающих подходящей избирательностью и чувствительностью, подобных методам окрашивания, используемым в микроскопии для выявления специфических тканей, или камерам Вильсона в физике элементарных частиц, позволяющим "материализовать" треки определенных событий.

3.4. Как разводить слайдеры

Трассировка окрашивает путь глайдеров, но короткий путь есть короткий путь и его нелегко увидеть. Можем ли мы подобрать "культуру", в которой глайдеры будут более долгоживущими? Можно ли "изолировать" глайдер? Используя редактор плоскости CAM, давайте сконструируем на экране маску, состоящую полностью из нулей, за исключением круглого острова единиц в середине, и запомним эту маску в буфере основного компьютера. Теперь запустим "жизнь", начиная с первичного бульона, примерно на сто шагов, когда условия зарождения глайдеров очень благоприятны. И тут мы применим операцию AND К сохраненной маске и конфигурации на экране/ для того чтобы сохранить только центральную часть картинки. Глайдеры, порожденные на берегу этого острова и движущиеся в окружающий океан, будут в состоянии проплыть длинную дистанцию беспрепятственно (рис. З.ЗЬ).

[1] Логическая операция AND действует на два двоичных входа и возвращает один двоичный выход; для того чтобы применить AND к двум конфигурациям, оператор применяется к месту нахождения клеток в обоих конфигурациях, вернее к соответствующим парам битов. CAM имеет средства для таких логических операций между буферами головного компьютера и конфигурациями CAM.



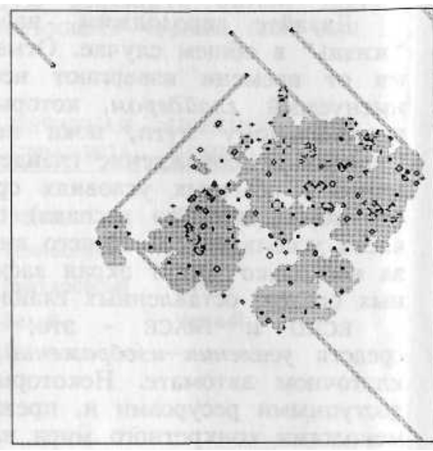


Рис. 3.3. Разведение глайдеров: (а) в общем случае глайдеры имеют короткую жизнь; (Ь) глайдеры, уплывающие от берега в открытый океан; стирая остров, мы получим чистую культуру глайдеров.

В случае удачного прогона мы можем обнаружить много глайдеров в одно и то же время! Когда это происходит, мы должны быть готовы с помощью новой маски (полностью состоящей из единиц, за исключением большого круглого отверстия из нулей) устранить то, что осталось от острова; теперь глайдеры смогут продолжить свой бег без риска разбиться о другие объекты. Они, конечно, могут спокойно сталкиваться *друг с другом* - но для чего еще необходимо собирать глайдеры?

Этот эксперимент по разведению глайдеров иллюстрирует, как заданием начальных условий, далеких от равновесия, можно увеличить возможности наблюдать ситуации, которые были бы в противном случае очень маловероятными. (Заметим, что точная форма и размер острова не имеют большого значения.)

Дальнейшая стадия понимания мира достигается тогда, когда мы можем определить на некотором уровне агрегирования ряд примитивных материалов и механизмов, которые можно использовать для построения структур, имеющих хорошо определенное, повторяющееся поведение. На этой стадии наука превращается в технологию. Чтобы строить механизмы произвольной сложности, мы должны уметь очень детально управлять начальными условиями в некоторой протяженной части мира.

На рис. 3.4а показана полная схема "глайдерной пушки" - машины, которая через регулярные промежутки времени выстреливает новый глайдер вдоль точно определенного пути; открытие этого устройства, равно как и других интересных явлений игры "жизнь", приписывается Б. Госперу. Наконец, на рис. 3.4Ь мы используем эту технологию для постановки эксперимента на сталкивающихся пучках, где два потока глайдеров вынуждены пересекаться и взаимодействовать.

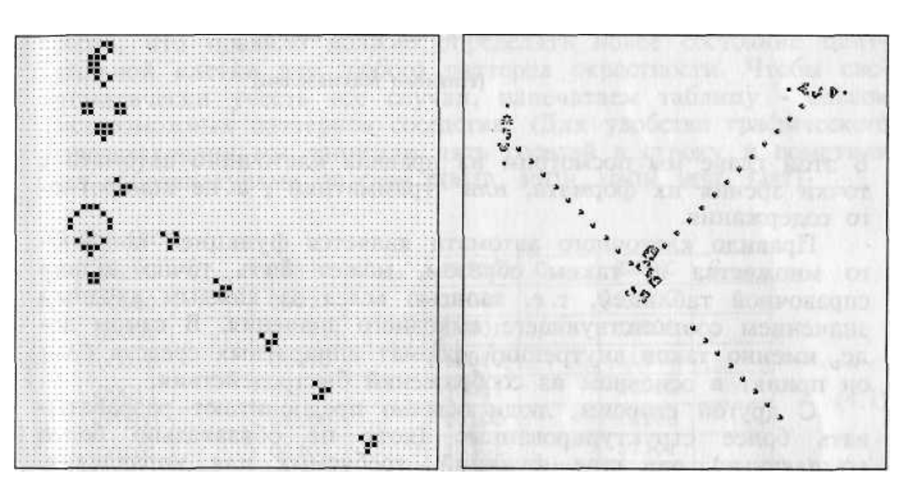


Рис. 3.4. (а) Глайдерная пушка (показано с увеличением). (Ь) Эксперимент со сталкивающимися пучками, использующий две глайдерные пушки.

Глава 4

ПРАВИЛА ИГРЫ

> Я не стану касаться республик... Здесь я перейду прямо к единовластному правлению и, держась намеченного выше порядка, разберу, какими способами государи могут управлять государствами и удерживать над ними власть.
>
> [Никколо Макиавелли]

В этой главе мы посмотрим на правила клеточного автомата с точки зрения их формата, или "грамматики", а не конкретного содержания.

Правило клеточного автомата является функцией конечного множества и, таким образом, может быть точно задано справочной таблицей, т. е. записью вслед за каждым входным значением соответствующего выходного значения. В самом деле, именно таков внутренний формат аппаратных средств CAM; он принят в основном из соображений быстродействия.

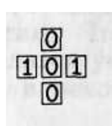
С другой стороны, люди обычно предпочитают разрабатывать более структурированные (хотя не обязательно более компактные) описания функций, требуемых для управления. Выбор подходящего языка описаний облегчает задание, распознавание или запоминание функций конкретного класса и (до некоторой степени) "понимание" их поведения.

4.1. Выбор вселенной

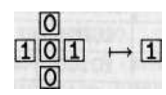
Машина клеточных автоматов стоит на нашем столе, ожидая, чтобы мы задали *правило* - закон, который будет править миром. С чего мы начнем?

Итак, насколько богатый у нас выбор? Иными словами, на что похоже правило?

Возьмите произвольную конфигурацию на экране, выберите клетку и посмотрите на паттерн битов, сформированный ее соседями. (В этой главе мы для простоты рассмотрим клеточные автоматы, использующие клетку с двумя состояниями и пятью соседями.) Мы можем, например, обнаружить



Может ли наше правило определить о или 1 для нового состояния клетки как "следствие" этого конкретного паттерна? Если в текущий момент мы придумываем правило, то вольны выбирать. Давайте выберем 1 и запишем этот факт в виде



Ясно, что правило должно определять новое состояние центральной клетки для любого паттерна окрестности. Чтобы систематически учесть все случаи, напечатаем таблицу - список всевозможных паттернов соседства. (Для удобства графического представления мы записали пять соседей в строку и пометили их аббревиатурами от слов CENTER, NORTH, SOUTH, WEST, EAST):

Правило: ...

EWSNC	Cnew	EWSNC	Cnew	EWSNC	vnew	EWSNC	'-'new
00000		01000		10000		11000	
00001		01001		10001		11001	
00010		01010		10010		11010	
00011		01011		10011		11011	
00100		01100		10100		11100	
00101		01101		10101		11101	
00110		OHIO		10110		11110	
00111		01111		10111		11111	

(4.1)

Чтобы задать правило, достаточно заполнить таблицу, записывая о или 1 в каждой из 32 позиций.

Сначала вы, возможно, захотели бы для каждого выбора бросить монету.[1] Для каждого правила, полученного таким образом, выполните несколько экспериментов,[2] начиная с различных видов начальных условий (случайное распределение, небольшое пятнышко на однородном фоне, лицо и т. д.). Чаще всего результаты будут весьма разочаровывающими; исходный паттерн будет хаотически расти, пока весь экран не будет заполнен "статикой", или сморщится и застынет в виде неподвижной конфигурации. Однако даже на этом бессистемном уровне экспериментирования некоторые правила продемонстрируют интересные особенности, и однажды, спустя некоторое время, вы получите настоящий самородок. Однажды авторы получили следующую таблицу:

[1] Этот процесс, конечно, может быть автоматизирован.

[2] В разд. 5.6 объясняется, как в CAM Forth может быть закодирована произвольная таблица.

Имя правила: HGLASS

EWSNC	vnew	EWSNC	Cnew	EWSNC	Cnew	EWSNC	vnew
00000	0	01000	0	10000	0	11000	0
00001	1	01001	0	10001	0	11001	1
00010	1	01010	0	10010	0	11010	0
00011	1	01011	1	10011	0	11011	0
00100	0	01100	0	10100	0	11100	0
00101	0	01101	0	10101	1	11101	1
00110	0	OHIO	0	ЮНО	0	11110	1
00111	0	ОНИ	0	10111	0	11111	1

(4.2)

Выполните это правило для различных начальных условий (начиная, в частности, с островка нулей в море единиц) и обратите внимание, какое большое разнообразие поведения укладывается в простую 32-разрядную таблицу. На рис. 4.1 и фото 1, 2 показано несколько примеров такого поведения.

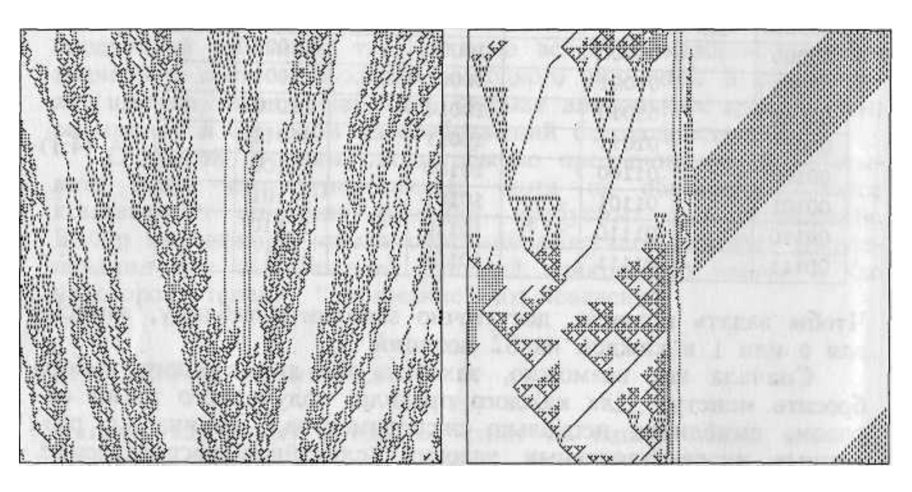


Рис. 4.1. Поведение HGLASS, если в качестве начальных ны: (а) случайные условия; (Ь) простой зародыш.

После случайного порождения нескольких сотен правил (трудно удержаться от синдрома игрока "Еще один разочек!") вы можете прийти к выводу, что это не очень эффективный путь конструирования интересных миров (или по крайней мере явно интересных миров).

Важнейшая черта хорошего стиля программирования - продуманная согласованность алгоритмов и структур данных. Аналогично, синтез полезных систем для клеточных автоматов проистекает из хорошего соответствия между выбором, сделан-

ным в пространстве "правил" (множестве справочных таблиц) и выбором, сделанным в пространстве "состояний" (множестве конфигураций). Каков размер пространства правил? Другими словами, сколькими различными способами можно заполнить указанную выше таблицу? Поскольку имеется $2^5 = 32$ позиций и 2 варианта на каждую позицию, то суммарное количество правил 2^{32}, или приблизительно 4 миллиарда! Для клеточных автоматов с двумя состояниями и девятью соседями ("формат" игры LIFE) это число вырастает до 2 в степени 2^9, или 2^{512}, что равно квадрату примерного числа элементарных частиц во Вселенной![1]

4.2. Словесная формулировка правил

Познакомившись с тем, как выглядит правило клеточного автомата с точки зрения *клетки,* мы теперь обсудим, как выразить это правило в удобном для *нас* виде. Задание отдельных элементов в большой таблице - утомительная работа, служащая источником ошибок; к тому же мы не хотим блуждать по огромному пространству правил без ощущения направления, структурного подхода к записи правил и осмысленных ограничений, вытекающих из природы рассматриваемой задачи.

То, что нам нужно, это язык для выражения правила в любых терминах, которые мы находим наиболее подходящими, и механизм для интерпретации наших запросов и трансляции их в справочную таблицу. На практике любой расширяемый язык программирования, если им достаточно легко пользоваться, пригоден для этих целей. Задачи самого нижнего уровня этого процесса трансляции могут поставляться конструктором машины как дополнительные, специализированные примитивы языка; задачи более высокого уровня могут принимать форму утилит общего назначения; наконец, проблемно-ориентированные потребности могут быть приняты пользователем во внимание за счет надлежащего объединения этих утилит или добавления к языку новых термов или конструкций.

Заметим, что, согласно принятому в CAM подходу, таблица данных, которая может использоваться миллиарды раз по ходу эксперимента, должна быть построена только однажды, перед началом эксперимента; эффективность процесса трансляции правила в справочную таблицу, таким образом, не влияет на быстродействие реализации клеточного автомата на ЭВМ. Поэ-

[1] CAM может работать с рядом различных окрестностей клеточного автомата, и для каждой окрестности внутренняя справочная таблица содержит до 8192 элементов; таким образом, количество правил, которые можно использовать в CAM, порядка $2^{8192} = 10^{2467}$! Более того, за счет композиции правил этот диапазон может быть существенно расширен.

3-1608

тому при представлении правила на выбранном языке можно расслабиться и сконцентрироваться на удобстве и ясности описаний, а не заботиться о программировании трюков, которые могли бы улучшить эффективность выполнения программы.

Простой пример проиллюстрирует маршрут, который ведет от описания клеточного автомата в словесной форме к запуску его на CAM. Правило, предложенное Эдвардом Фредкином из МТИ в начале эры клеточных автоматов [3], утверждает, что состояние клетки будет "соответствовать *четности* ее окрестности", т. е. что она станет живой или мертвой в зависимости от того, содержит ли окрестность в текущий момент нечетное или четное количество живых клеток. Более формально,

$$\mathrm{CENTER}_{new} = \mathrm{CENTER\ ф\ NORTH\ ф\ SOUTH\ 8\ WEST\ ф\ EAST} \quad (4.3)$$

(где © означает сумму по модулю 2). Значения этого выражения для всех возможных значений его аргументов дает следующая справочная таблица:

Правило: PARITY

EWSNC	Cnew	EWSNC	Cnew	EWSNC	Cnew	EWSNC	Cnew
00000	0	01000	1	10000	1	11000	0
00001	1	01001	0	10001	0	11001	1
00010	1	01010	0	10010	0	11010	1
00011	0	01011	1	10011	1	11011	0
00100	1	01100	0	10100	0	11100	1
00101	0	01101	1	10101	1	11101	0
00110	0	OHIO	1	10110	1	11110	0
00111	1	ОНИ	0	10111	0	11111	1

(4.4)

но это как раз то, что мы хотим сделать автоматически, даже не заботясь о том, чтобы взглянуть на результат. В CAM Forth мы могли бы закодировать выражение (4.3) так:

```
N/MOORE
                                          PARITY
CENTER NORTH SOUTH WEST EAST
                XOR  XOR  XOR  XOR
                               >PLNO
```

Описание N/MOORE утверждает, что для этого эксперимента мы планируем использовать окрестность *Мура,* в которой имеются слова соседства CENTER . . . EAST.[1] Двоеточие указывает Forth,

[1] Каждая окрестность CAM сопровождается собственным множеством подходящих слов соседства, как объясняется в гл. 7. Соответствующий "монтаж" машины позволяет клетке получать информацию от этих соседей.

что мы собираемся добавить к его словарю новое слово, а именно PARITY. Тело определения, заканчивающееся литерой ';', является простым выражением в обратной польской записи; XOR - это слово Forth для операции сложения по mod 2. Слово PARITY никогда не будет вызываться нами непосредственно; вместо этого оно будет передаваться как аргумент другому слову MAKE-TABLE, которое будет знать, что с ним делать. Затем будут выполнены следующие действия:

> HAKE-TABLE пробежит весь список всевозможных конфигураций соседства; для каждой конфигурации оно назначит предопределенным словам CENTER . . . .EAST значения, которые имеют в этой конфигурации соответствующие соседи, и вызовет PARITY.
>
> Всякий раз, когда PARITY вызывается с новым набором значений соседей, оно вычислит соответствующее новое значение центральной клетки и запомнит его в подходящем элементе таблицы, связанном с плоскостью О (>PLNO знает, где располагается эта таблица, и действует как посредник).
>
> В конце концов, MAKE-TABLE перенесет всю таблицу в CAM, где она запомнится в памяти с быстрым доступом, подготовленную должным образом для того, чтобы пользоваться ею в ходе эксперимента.

Теперь нам осталось только "запустить" CAM. Зададим на экране начальную конфигурацию, скажем небольшой квадрат в середине, и нажмем клавишу RUN. Согласно правилу PARITY, квадрат будет быстро эволюционировать, пульсируя и создавая узоры типа "персидский ковер", как на рис. 4.2.

Одним совсем не очевидным из справочной таблицы свойством PARITY, которое совершенно отчетливо обнаруживается структурным описанием (4.3), является его *линейность.*[1] Из этого факта можно формально вывести важные заключения; например, волны не искажаются в результате прохождения друг сквозь друга. Другое свойство, которое может быть формально выведено из (4.3), состоит в том, что для любой исходной фигуры на однородном фоне эта фигура будет обнаружена воспроизведенной в пяти копиях после соответствующего промежутка времени (а позже в двадцати пяти копиях и так далее).

[1] Возьмите любые две конфигурации и проследите их поведение в двух отдельных экспериментах, скажем до ста шагов, а затем сложите две конечные конфигурации вместе (сложите соответствующие позиции по mod 2). Результат будет таким же, как если бы две конфигурации были сначала сложены, а затем прошли сто шагов в одном эксперименте.

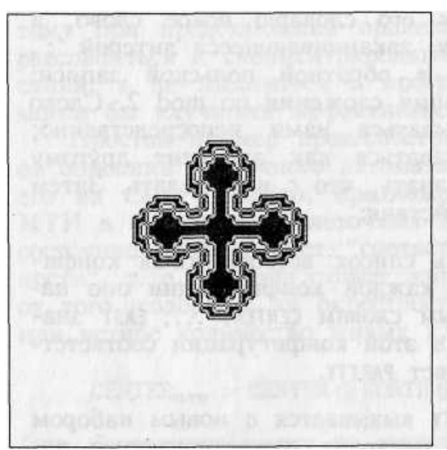
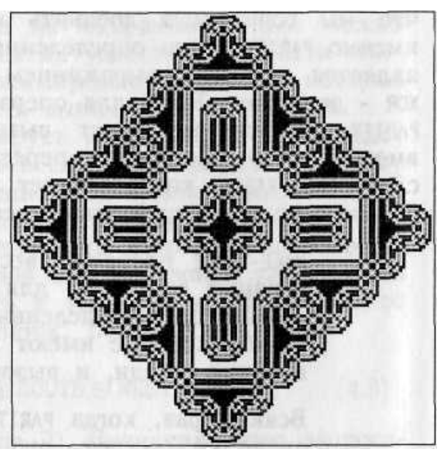



Рис. 4.2. Узоры, созданные правилом PARITY, если начать с квадратного образца 32x32, после приблизительно (а) пятидесяти и (Ь) ста шагов.

Хотя не следует поддаваться искушению приписывать чудодейственную силу форме записи законов, однако удачно выбранная форма часто способствует пониманию существенных особенностей поведения системы.

Алгоритмы часто используются как сжатые представления больших таблиц данных, обладающих определенной регулярностью. В нашем положении - когда таблицы содержат самое большее несколько тысяч битов - преимущество использования алгоритмов, а не таблиц в описании правил, состоит не столько в краткости, сколько в структурной ясности. Например, строку из 32 бит, составленную из столбцов c_{new} табл. (4.4), в шестнадцатиричных цифрах можно записать как 6DD6D66D, и эту строку можно было бы далее использовать в качестве *канонического* имени правила; однако кто бы распознал в

```
088E8EE38EE3E3308EE3E330E3303000
8EE3E330E3303000E330300030000000
8EE3E330E3303000E330300030000000
E3303000300000003000000000000000
```

правило LIFE из гл. 3?

Систематические схемы присвоения имен - в которых правило определенного класса может быть восстановлено по его имени - иногда используются, но лишь в специальных ситуациях (см. [55, 66, 71, 38]).

Часть II. Возможности

Глава 5

НАШИ ПЕРВЫЕ ПРАВИЛА

Вот пятнышко.
Вот пятнышко бежит.
Беги, пятнышко, беги!
Беги, беги, беги...
Пятнышко может бегать быстро.

[Детская книжка]

В этой главе мы обсудим правила клеточного автомата, которые самым простым способом используют наиболее доступные возможности САМ.

Даже этот простой класс правил уже способен дать богатое разнообразие явлений, применимых к множеству моделей. Кроме того, хорошее знакомство с основными строительными блоками позволит легче работать с более сложными конструкциями, которые будут введены в следующих главах. (Как и в обычном программировании, первичные средства САМ могут быть организованы и структурированы с помощью многих концептуальных методов и превращены в иерархию инструментов, пригодных для определенных задач.)

Все обсуждаемые в этой главе правила будут использовать единственную плоскость битов и окрестность Мура (см. 7.3.1), состоящую из центральной клетки и восьми ближайших соседей.

5.1. Неограниченный рост

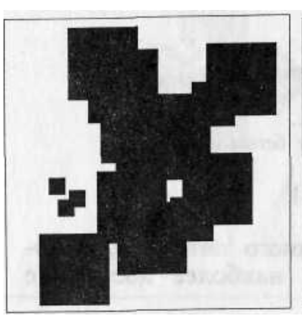
Давайте очистим экран САМ (все нули) и запустим следующее правило:

```
                                  : SQUARES
    N.WEST   NORTH    N.EAST
    WEST     CENTER     EAST
    S.WEST   SOUTH    S.EAST
OR OR OR OR OR OR OR OR  >PLNO ;
```

Экран останется белым. Однако как только мы поместим на экран одну единицу, из этого "зародыша" на нем возникнет черный квадрат, растущий с постоянной скоростью; несколько секунд спустя черная область заполнит экран. Если рассеять

по экрану больше зародышей, то возникнет большее число квадратов, которые в процессе роста будут перекрывать друг друга (рис. 5.1а).

Это простой пример монотонного[1], неограниченного роста.



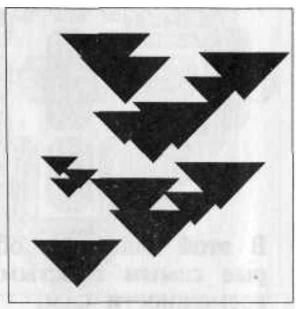


Рис. 5.1. Монотонный рост зародышей: (а) SQUARES, (b) DIAMONDS и (с) TRIANGLES.

В приведенном выше правиле SQUARES выражение с операцией OR использует девять соседей, размещенных в квадрате размером 3x3, в середине которого находится центральная клетка; результатом является "квадратный" рост. Аналогично, правило DIAMONDS

```
                    : DIAMONDS
     NORTH
WEST CENTER EAST
     SOUTH
     OR OR OR OR >PLNO ;
```

использует расположение пяти соседей[2] в форме ромба, а результатом является "ромбовый" рост (рис. 5.1Ь). Прежде чем пытаться делать обобщения в этом направлении, попробуем конфигурацию, в которой симметрия четвертого порядка нарушаема; например, треугольник, вершина которого обращена на север, как в правиле

```
                    : TRIANGLES
     NORTH
WEST CENTER EAST
         OR OR OR >PLNO :
```

[1] То есть однажды включенная клетка остается включенной постоянно.

[2] Расположение текста в этом фрагменте программы на языке Forth, конечно, не имеет никакого значения и было выбрано только для ясности.

Результатом роста являются треугольники, вершиной обращенные на *юг*\ В общем случае любой элемент выражения с операцией OR, расположенный в окрестности в определенном направлении от центра, будет вызывать рост в *противоположном* направлении. Это не удивительно, поскольку соседи действуют как *источники* информации для центральной клетки.

Чтобы глубже уяснить это положение, заметьте, что правило, которое используется САМ, когда вы нажимаете клавишу с обращенной вниз стрелкой с целью сдвинуть все содержимое экрана к югу, есть

```
            : SHIFT-SOUTH
    NORTH >PLNO;
```

То есть информация, которая движет на юг, должна прийти с *севера,* где и следует ее искать.

Световой конус события (обновления клетки) состоит из всех событий в прошлом, которые могут влиять на конечный результат, и всех событий в будущем, которые могут подвергнуться воздействию этого результата. В клеточном автомате *скорость света* (максимальная скорость распространения информации) может быть различной в различных направлениях и, таким образом, форма светового конуса пространства-времени клеточного автомата может в действительности быть скошенной пирамидой. В любом случае клетки, которые могут подвергнуться воздействию за один шаг (поперечное сечение светового конуса будущего), располагаются в зеркально-симметричном отражении (а не в идентичной копии) ее окрестности (поперечное сечение светового конуса прошлого) относительно точки.

5.2. Ограниченный рост

В правиле SQUARES предыдущего раздела черная область растет "с максимально возможной скоростью" (т. е. со скоростью света), потому что клетка оживает, как только она обнаруживает живую клетку в своей окрестности. Можно сделать процесс роста более избирательным, ограничивая число допустимых состояний, которые может принимать клетка. Например, в следующем правиле клетка включается, лишь если она обнаруживает *в точности* одну живую клетку среди восьми своих соседей, а иначе должна остаться неизменной. (Слово 8SUM использовалось в примере LIFE из разд. 3.1.)

```
            : 1-0UT-0F-8
    8SUM 1 - IF
```

```
        1 ELSE
   CENTER THEN
        >PLN0 ;
```

Получающаяся в результате роста конфигурация намного разреженней, чем в случае SQUARES. Обратите внимание на регулярный фрактальный характер этой конфигурации.

Целый набор правил можно получить вариациями на тему "подсчета", используя следующую схему (для краткости мы определим слово и - для операции "без изменений" - как аббревиатуру CENTER):

```
                  : U
            CENTER ;
                  : LICHENS
   8SUM { U U U 1 U U U 1 1 }
             >PLNO ;
```

Конструкция в скобках - *оператор выбора,* содержащий один элемент для каждого возможного значения своего аргумента (8SUM - 0,...,8). Здесь эта конструкция используется как простая таблица решений, а 1 в позиции *n* означает, что клетка включится, если включены *n* соседей; и означает, что она останется неизменной. В приведенном выше примере, рост допускается, только когда число живых соседей равно 3, 7 или 8; для начального роста необходим зародыш по крайней мере из трех клеток; при этом в зависимости от формы зародыша рост может продолжаться неограниченно или остановиться спустя некоторое время (рис. 5.2Ь). Правило INKSPOT из разд. 1.2 работает аналогично.

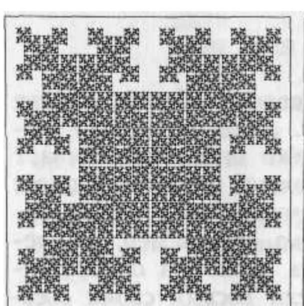
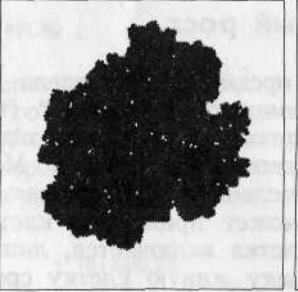
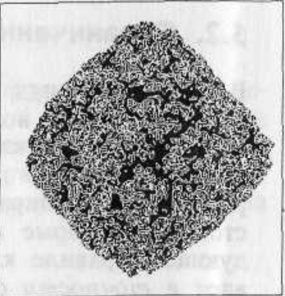




Рис. 5.2. Ограниченный рост: (а) 1-OUT-OF-8, (b) LICHENS, (c) LICHENS-WITH-DEATH

5.3. Конкурентный рост

Бели в указанной выше таблице заменить некоторые элементы нулями, то возникнут ситуации, где белая часть будет снова увеличиваться за счет черной. В этих случаях обычно очень трудно предсказать результаты долговременного развития; в действительности же из теории вычислений хорошо известно, что исключительно простые механизмы конкурентного роста способны поддерживать универсальные в вычислительном отношении процессы. Следующее правило получается из приведенного выше правила LICHENS заменой одного элемента на о:

```
                  : LICHENS-WITH-DEATH
8SIJM  { U U U 1 O U U 1 1 }
              >PLNO ;
```

Получающийся в результате характер роста совершенно иной.

Игра LIFE, обсужденная в гл. 3, принадлежит к этому классу правил и могла бы в действительности быть определена как

```
                  : LIFE
  8SUM { 0 0 U 1 0 0 0 0 0 }
              >PLNO ;
```

5.4. Правила голосования

Правила, обсужденные в двух предыдущих разделах, являются правилами *подсчета,* в которых поведение клетки зависит только от того, *сколько* соседей находится в данном состоянии, безотносительно к их детальному пространственному расположению. Дальнейшая специализация возникает, когда вклад каждого соседа интерпретируется как "голос" в пользу определенного результата; любое число голосов свыше определенного порога приведет к этому результату.

В следующем правиле клетка будет следовать состоянию большинства ее соседей. Голос самой клетки учитывается словом 9SUH, аналогичным 8SUM; это ведет к диапазону из 10 различных возможностей 0,1,...,9, который может быть поровну разделен установлением порога между четырьмя и пятью ("простое большинство").

```
                  : MAJORITY
9SUM { 0 0 0 0 0 1 1 1 1 1 }
              >PLNO ;
```

Почти очевидно, что в областях, в которых черные имеют даже незначительное большинство, будет наблюдаться тенденция к консолидации этого большинства; аналогично для белых. Неочевидно другое: что произойдет, когда встретятся два цвета? Будет ли граница резкой или размытой? Будет ли она устойчивой? Будет ли она стремиться выпрямиться или стать более извилистой? Мы обнаружим, что на поведение границы могут существенно повлиять небольшие изменения в этом правиле.

В случае правила MAJORITY, если начать с экрана со случайным распределением нулей и единиц, то через несколько шагов экран сам перестроится во взаимно проникающие черные и белые области, поддерживаемые устойчивыми "альянсами". Если начать с низкого процента единиц, то к концу белая область будет преимущественно связной, оставляя черные острова, окруженные белым океаном; ситуация постепенно меняется на противоположную, по мере того как начальная доля единиц увеличивается (см. рис. 5.3а). Модели этого вида полезны при изучении явлений *зародышеобразования* и *перколяции*.

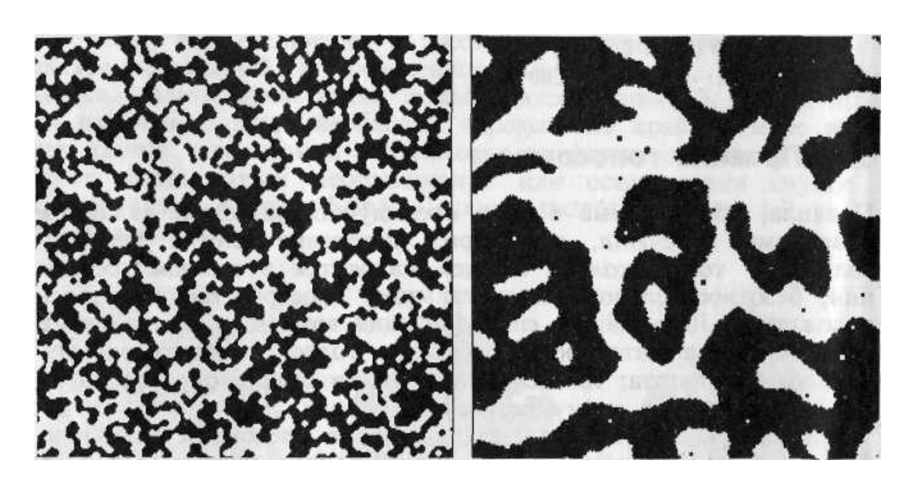


Рис. 5.3. Правила голосования, если начать с 50% случайно расположенных единиц, (а) При простом большинстве получаем устойчивые, весьма фрагментарные области. (Ь) Большинство с отжигом (ANNEAL) приводит к непрерывному объединению областей; здесь действует режим ECHO.

Интересный вариант этого правила построил Вишняк [67]. Обмен двух элементов таблицы в непосредственной близости от порога, приводящий к правилу

```
                    : ANNEAL
9SUM { 0 0 0 0 1 0 1 1 1 1 }
                    >PLNO ;
```

способствует перетасовке границы между черными и белыми областями, где большинство минимально.

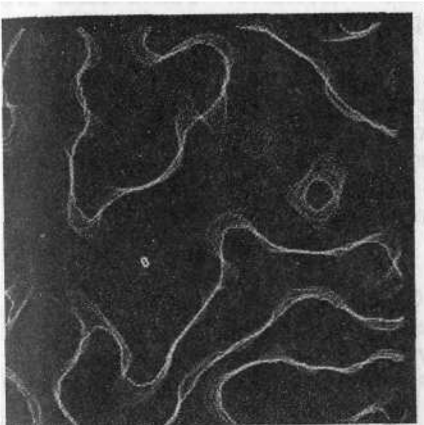
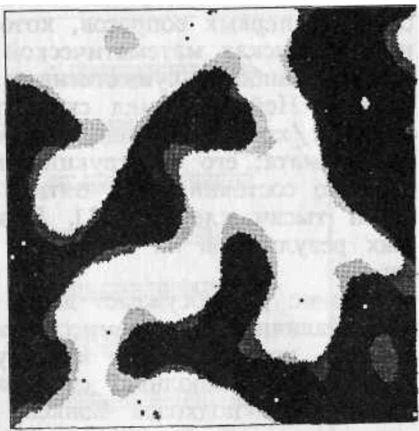



Рис. 5.4. Поверхностное натяжение в правиле ANNEAL: (а) двойная экспозиция с интервалом в 400 тактов; заливы наполняются, а мысы размываются; (Ь) длительная экспозиция, регистрирующая блуждание фаниц.

Суммарный эффект состоит в постепенном *"отжиге"* доменов по прошествии длительного времени; каждая клетка начинает вести себя так, как если бы голосование отражало состояние не только ее ближайших соседей, но, в убывающей степени, и состояния клеток, все более и более далеких от нее. Области формируются как и прежде, но теперь границы находятся в состоянии непрерывного брожения; каждая клетка может, так сказать, "чувствовать" кривизну своей окрестности и будет динамически регулировать свое состояние так, чтобы сделать границу более прямой (рис. 5.3Ь): "заливы" заполняются, а "мысы" размываются, как показано на рис. 5.4а. На макроскопическом уровне детальная механика правила смазывается, а то, что остается, является хорошей моделью поверхностного натяжения. При этом границы ведут себя как натянутые мембраны, которые испытывают натяжение, пропорциональное их кривизне (рис. 5.4Ь и фото 3). В части III мы обсудим другие варианты, в которых дискретный микроскопический механизм обеспечивает хорошие модели для известных континуальных явлений.

5.5. Компьютер Бэнкеа

В конце разд. 3.4 мы отметили, что в мире, определенном данным правилом клеточного автомата, можно найти "материалы" и "механизмы", которые могут быть собраны в сложные устройства согласно определенному плану и цели. Можно ли построить компьютер на клеточном автомате? Фактически это один из первых вопросов, который был поставлен фон Нейманом в поисках математической "вселенной", способной поддерживать наиболее существенные черты жизни. В конечном счете фон Нейман сумел сконструировать универсальный вычисляющий/конструирующий "робот", живущий внутри клеточного автомата; его конструкция использовала клетки с двадцатью девятью состояниями и пятью соседями и занимала несколько сотен тысяч клеток [68]. Позже Кодд [11] достиг аналогичных результатов на клеточном автомате с восемью состояниями.

Бэнкс [3] обсуждает даже более простые решения, используя различные компромиссы между числом состояний, числом соседей, компактностью конструкции и текстурой среды, в которой робот выполняет свои задачи. Здесь мы приведем простейший из подходов Бэнкса, пригодный для построения вычислительных схем произвольной сложности.

Правило Бэнкса полностью определяется следующими тремя элементами

```
  1               0               1
0 1 1  ↦ 0     1 0 1  ↦ 1     1 0 1  ↦ 1
  0               1               1
```

(с учетом того, что четыре повернутые по часовой стрелке варианта каждого элемента приводят к тому же результату); для всех других элементов таблицы правило указывает "без изменения" для центральной клетки. С учетом расположения единиц в поле нулей правило в основном гласит - "заполняйте впадины, выгрызайте углы". В CAM Forth, используя, как и прежде, U вместо "без изменений", получим

```
                                      : CORNER?
    NORTH SOUTH = IF U ELSE 0 THEN ;
                                      : BANKS
       NORTH SOUTH WEST EAST + + +
        { U U CORNER? 1 1 } >PLNO  ;
```

Если число соседей в состоянии единица (без учета центральной клетки) равно двум, то мы должны решить, находятся ли они на прямой линии или под углом 90°; в последнем случае мы заставляем угловую клетку принять состояние 0. Это единственный случай, когда может возникать "смерть".

Если мы запустим это правило, начиная со случайных начальных условий, то после нескольких шагов получим приятную, но непонятную текстуру на рис. 5.5а; в нескольких местах мы можем различить небольшие участки активности с двигающимися вперед и назад сигналами. Можем ли мы "приручить" эту активность и направить ее на более полезные задачи?

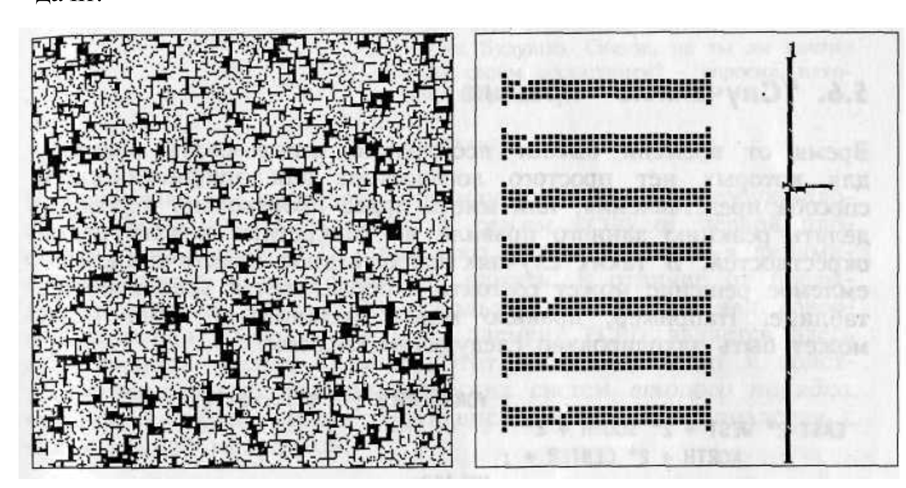


Рис. 5.5. Правило BANKS, (а) Конфигурация, развившаяся из случайных начальных условий, (о) Искусственные устройства: распространение сигнала по проводу (слева, увеличено) и столкновение двух потоков сигналов, генерируемых часами с различными периодами.

Оказывается, что сигналы можно легко заставить распространяться по таким "проводам". Действительно, если вы вырежете наклонную ступеньку на краю сплошной черной области, то она будет перемещаться на одно положение за каждый шаг; при противоположном наклоне она будет двигаться в противоположном направлении. Для поддержания сигнала черная область должна быть глубиной в три клетки - это и будет наш *провод.* В левой половине рис. 5.5Ь показано распространение сигнала; достигнув конца провода, он исчезает.[1]

Набравшись терпения (подробности есть в [3]), можно показать, что сигналы можно заставить заворачивать за угол, пересекаться и разветвляться. Наконец, можно построить "часы", которые генерируют поток импульсов с регулярными интервалами, во многом подобные "глайдерной пушке" из разд.

Углы на свободном конце провода должны быть защищены небольшим зубцом, как показано на рисунке, или они будут выгрызены.

3.4, и вентиль с двумя входами и одним выходом, который вычисляет логическую функцию *a* AND NOT *b*. Это все, что нужно для построения универсального компьютера. В правой части рис. *5.5Ъ* показаны часы, испускающие по проводу сигналы с периодом 16 тактов, и другие часы с периодом 32 такта, перехватывающие и уничтожающие каждый второй импульс.

5.6. "Случайные" правила

Время от времени бывает необходимо использовать правила, для которых нет простого логического или арифметического способа представления, или иметь право произвольно переопределять реакцию данного правила на конкретные конфигурации окрестностей. В таких случаях единственное практически приемлемое решение может состоять в обращении к заполненной таблице. Например, правило H6LASS, заданное в табл. (4.2), может быть закодировано следующим образом:

```
                              : VONN-INDEX (-- 0 . . . .31)
  EAST 2* WEST + 2* SOUTH + 2*
          NORTH + 2* CENTER + ;
                              : HGLASS

VONN-INDEX  { 0 1 1 1 0 0 0 0
              0 0 0 0 0 0 0 0
              0 0 0 0 0 1 0 0
              0 1 0 0 0 1 1 1 }
                         >PLNO ;
```

где VONN-INDEX нумерует от 0 до 31 тридцать две конфигурации окрестности фон Неймана в том же порядке, как они представляются в (4.1).

Глава 6

ДИНАМИКА ВТОРОГО ПОРЯДКА

- Я Святочный Дух Прошлых Лет!
- Я - Дух Нынешних Святок!
- Дух Будущих Святок, не ты ли почтил меня своим посещением? - спросил, наконец, Скрудж.

[Диккенс]

В этой главе мы позволяем правилу взглянуть на вторую плоскость битов - установленную в режим ECHO от первой плоскости, как в разд. 3.2, - для того, чтобы добавить в динамику небольшую кратковременную память. В отличие от предыдущей ситуации, где ECHO использовалось только для совершенствования изображения LIFE, элемент ECHO является здесь существенной частью правила. Этот подход приводит к конструированию множества динамических систем *второго порядка,* включая системы, демонстрирующие замечательные аналогии с ньютоновской механикой.

6.1. Возбуждение нейронов: правило с тремя состояниями

При введении LIFE мы коснулись аналогии с популяционной динамикой; в результате мы получили содержательную терминологию, отражающую эту аналогию, но там не было претензий на моделирование реальных систем. В подобном же смысле следующее правило может быть интерпретировано как описывающее динамику информационных паттернов в "мозгу", состоящем полностью из нейронов без аксонов: нейроны плотно упакованы и поддерживают связь за счет непосредственного соприкосновения со своими соседями.[1]

Правило описывает клетки, имеющие *три* состояния: о ("готова"), 1 ("возбуждение") и 2 ("рефрактерное"). Готовая клетка возбудится, если в точности два из восьми соседей находятся в состоянии возбуждения; после возбуждения она перейдет в рефрактерное состояние, в котором она нечувствительна к раздражению, и, наконец, возвратится в состояние готовности. В табличной форме эти правила имеют вид

[1] Это правило было предложено Б. Сильверманом; "нейронная" интерпретация наша.

Ш •-> Ш (только когда два соседа находятся в состоянии 1)

ва-ва **(6Л)**

Используя две плоскости битов, как в разд. 3.2 (сопровождение эхом), мы закодируем состояния нейрона следующим образом:

СТАТУС КЛЕТКИ (МОЗГ)	СОСТОЯНИЕ	ПЛОСКОСТЬ 0	1
готова	0	0	0
возбуждение	1	1	0
рефрактерное	2	0	1
не используется	3	1	1

(6.2)

Поскольку, согласно этой схеме кодирования, в плоскости О зарегистрировано, какие клетки находятся в состоянии возбуждения, а в плоскости 1 - какие клетки находятся в рефрактерном состоянии, то легко выразить общее правило (6.1) посредством двух отдельных составляющих правила, для плоскости 0 и для плоскости 1.

Заметьте, что клетка будет иметь рефрактерное состояние сразу после состояния возбуждения; поэтому правило для плоскости 1 есть просто ECHO (обсужденное в разд. 3.2 и для удобства повторенное ниже), которое помещает копию плоскости 0 в плоскость 1 с задержкой на один шаг:

```
              : ECHO
  CENTER >PLN1 ;
```

Что касается плоскости 0, то мы определим сначала слово, которое возвращает 1, только если имеется соответствующее значение STIMULUS, И слово, которое сообщает нам, находится ли клетка в состоянии READY. Новое состояние клетки определяется затем взятием операции AND OT двух этих условий:

```
                          : READY   ( -- 0|1)
      CENTERS { 1 0 0 0 } ;
                          : STIMULUS  ( - 0|l)
8SUM { 0 0 1 0 0 0 0 0 0 } ;
                          : BRIAN'S-BRAIN
  STIMULUS READY AND >PLNO ;
```

В коде слова **READY** мы использовали новое "слово соседства", а именно **CENTERS.** Это слово выражает совместное состояние **CENTER** и **CENTER'** как единственную переменную, принимающую значения 0, 1, 2 и 3 - как в табл. (6.2).[1]

Если бы даже из-за небрежности экспериментатора начальная конфигурация содержала несколько клеток в состоянии 3 ("не используется"), то, согласно данному правилу, аномалия исчезнет за один шаг.[2]

Интересный аспект BRIAN'S-BRAIN состоит в том, что не существует статических структур. Вся активность является быстротечной и непрерывно подкрепляется взаимной стимуляцией возбужденных паттернов. Рис. 6.1a - моментальный снимок - вряд ли может должным образом отразить то, что видно на экране.

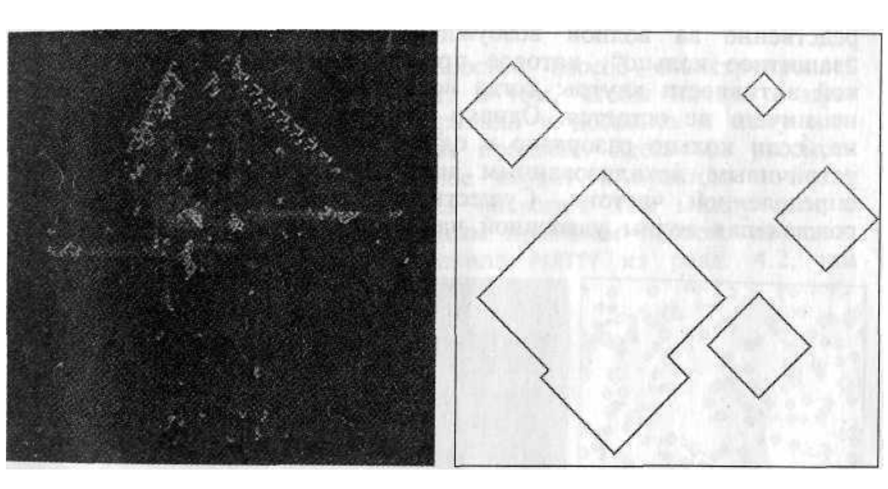


Рис. 6.1. (а) Краткая временная выдержка в **BRIAN'S-BRAIN.** (b) правило GREENBERG: изолированные точки превращаются в кольца, которые сливаются в еще большие кольца.

Однажды разученный прием уже несложно применить к множеству различных ситуаций. В сочетании с ECHO слово READY может быть использовано как "ингибирующая" компонента для *любого* правила, работающего в плоскости 0. Мы можем, например, взять правило DIAMONDS (разд. 5.1), которое само по себе приводит к тривиальному, неограниченному росту, и записать

[1] Все слова соседства, имеющиеся в CAM Forth, обсуждаются в гл. 7. В любом случае **CENTERS - CENTER + 2 x CENTER'.** Конечно, мы могли бы определить **READY** с помощью операции OR от слов **CENTER** и **CENTER';** однако здесь нам хотелось бы избавить читателя от тонкостей логико-арифметической конверсии (см. разд. А. 14).

[2] Во время этого шага клетки будут вести себя как возбужденные.

```
                                    : GREENBERG
       CENTER NORTH SOUTH WEST EAST      \ эта часть является
              OR OR OR OR                \ правилом DIAMONDS
                    READY AND >PLNO ;
```

Это правило, некоторые аспекты которого были изучены Гринбергом и Гастингсом [21], имеет несколько "видов" активности, зависящих от начальных условий. Единственная точка вырастет в ромбовидный волновой фронт (с пустой внутренностью), перемещающийся со скоростью света; когда два таких "кольца" сталкиваются, их общая граница уничтожается, что в результате дает полую структуру большего размера с непрерывной границей (рис. 6.1Ь). Если плоскость 1 изначально пуста, волна подавления в этой области следует непосредственно за волной возбуждения в плоскости 0, создавая "защитное кольцо", которое препятствует проникновению всякой активности внутрь: когда возбуждение закончено, на экране ничего не остается. Однако длительная активность возможна, если кольцо разорвано в одной точке: эта щель становится устойчивым локализованным источником периодических волн определенной частоты. Существует также конфигурация, порождающая волны удвоенной частоты.

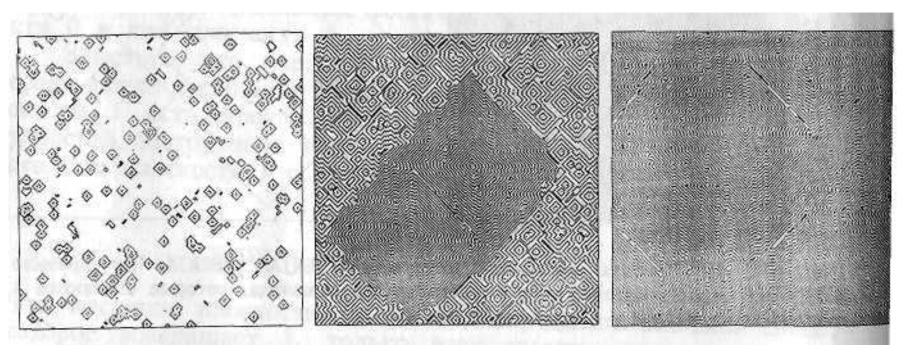


Рис. 6.2. Правило GREENBERG. Начальное, промежуточное и установившееся поведение, наблюдаемые при запуске конфигурации с 3% единиц, расположенных случайно в одной плоскости, и с другим случайным расположением 3% единиц в другой плоскости.

Хороший способ экспериментирования с этим правилом состоит в том, чтобы использовать в качестве начальной конфигурации паттерн, содержащий небольшую долю единиц, случайно размещенных в плоскостях 0 и 1. Медленно меняя эту долю от запуска к запуску, можно получить различные формы поведения. На рис. 6.2 показана серия кадров, полученная из начального состояния с 3% единиц.

6.2. Задний ход

Существенное свойство правил предыдущего раздела состоит в том, что новая конфигурация плоскости 0("будущее") конструируется с учетом как текущей конфигурации этой плоскости ("настоящее"), так и предыдущей ее конфигурации ("прошлое"); последнее было сохранено ECHO в плоскости 1. Другими словами, если мы рассмотрим последовательность конфигураций

$$c^{t-2} \quad c^{t-1} \quad c^{t} \quad c^{t+1} \quad c^{t+2} \qquad (6\ 3)$$

через которые проходит система с течением времени, то каждая конфигурация определяется двумя предыдущими. Это характеризует систему *второго порядка*.

Как упоминалось выше, простой способ сконструировать систему второго порядка состоит в том, чтобы начать с простого (или *первого порядка)* правила и добавить к нему выражение, содержащее ссылку на прошлое. Здесь мы рассмотрим пример, в котором прошлое входит в правило в очень простом виде, а именно берется операция XOR между ним и значением, порожденным исходным правилом первого порядка. Отправляясь, например, от правила PARITY из разд. 4.2, мы можем сконструировать новое правило

```
                         : PARITY-FLIP
CENTER NORTH SOUTH WEST EAST
             XOR XOR XOR XOR
                CENTER' XOR
                    >PLNO ;
```

в котором значение, возвращаемое изначально PARITY, в дальнейшем подвергается операции XOR с CENTER' (текущее значение в плоскости 1 является предыдущим значением плоскости O, так как действует ECHO), перед тем как использоваться в качестве следующего значения для плоскости 0.

Согласно новому правилу, прошлое просто копируется в будущее либо "как есть" (когда значение, возвращаемое исходным правилом, есть о), либо после *образования его дополнения* (когда это значение 1). Часть первого порядка этого правила может быть представлена как "силовое поле", которое состояние системы должно пересекать при движении из прошлого в будущее; в зависимости от значения этой силы это состояние может появиться вновь неизменным или оказаться "перевернутым".

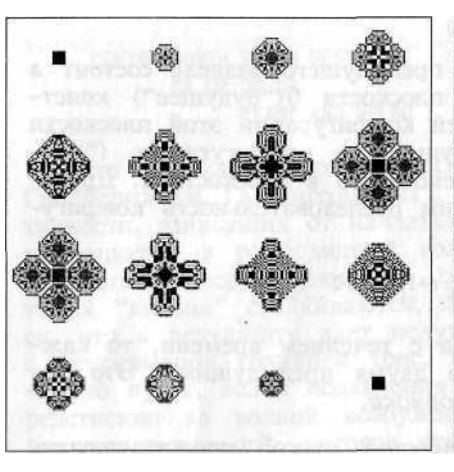
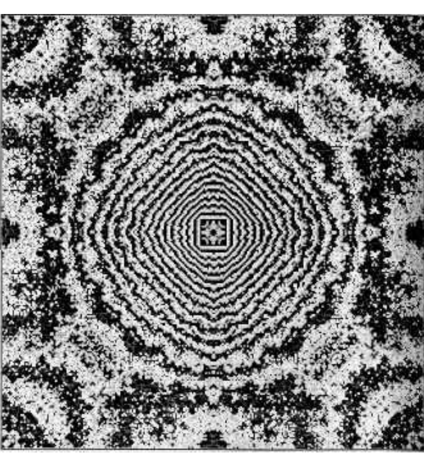



Рис. 6.3. (a) Восемь стадий в эволюции PARITY-FLIP с последующей идентичной эволюцией во времени обратно, (b) TIME-TUNNEL: эволюция от начальной конфигурации в виде квадрата после тысячи шагов.

В качестве первого эксперимента с этим правилом для простоты начните с маленького квадрата единиц в плоскости O и пустой конфигурации в плоскости 1 и выполните тысячу шагов (рис. 6.3a). Характер динамического поведения совершенно отличен от поведения исходного варианта PARITY: на всех уровнях рост чередуется с устойчивым отступлением, а не сменяется внезапным коллапсом. Переставим теперь содержимое двух плоскостей битов[1] и выполним еще тысячу шагов; *фильм пойдет в обратном порядке,* и вы возвратитесь в исходную начальную позицию. Продолжите ваше движение еще на несколько шагов - теперь вы путешествуете в (пока еще не исследованное) прошлое системы! Снова переставьте прошлое и настоящее, и система опять начнет двигаться вперед по времени. PARITY-FLIP является *обратимой во времени* системой, и мы имеем доступ к ее "заднему ходу".

Хотя зрительные впечатления схожи, этот эксперимент с обратимостью совершенно отличен от обычной "трюковой съемки". В реальном мире можно создать фильм, изображающий, как разбросанные взрывом обломки сами собираются и образуют, скажем, дом; но это можно сделать лишь ценой запоминания в фильме *всех отдельных кадров* прямого процесса, а затем демонстрацией их в обратном порядке. Во время

[1] Команды для свопинга плоскостей, конечно, непосредственно доступны в CAM.

демонстрации фильма нет причинной связи между отдельными кадрами (они были расположены в этом конкретном порядке монтажником, который мог бы выбрать и другой порядок с еще более удивительными результатами). В нашем эксперименте, напротив, мы запоминаем только последнюю пару кадров и затем даем правилу возможность восстановить в реальном времени историю прошлого системы в соответствии с принципами строгой причинности.

Более полное рассмотрение обратимости в клеточных автоматах будет дано в гл. 14; в частности, теоретическое обоснование приведенного выше примера дается в разд. 14.2.

6.3. Непроницаемый барьер

Мы выполним эксперимент с другим обратимым правилом второго порядка, а именно

```
                                  : TIME-TUNNEL
CENTER NORTH SOUTH WEST EAST
    + + + + { 0 1 1 1 1 0 }       \ возвращает единицу, если
                                  \    нет полного совпадения
          CENTER' XOR >PLNO ;     \ операция XOR с прошлым
```

(ECHO вновь является правилом для плоскости 1). Небольшой квадрат из троек[1] в середине будет действовать как постоянный источник волн. Так как наше пространство клеточного автомата замкнуто (см. разд. 2.2.5), то эти волны будут распространяться по всему кругу - или, точнее, по всему *тору* - и возвращаться, сталкиваясь и порождая целый ряд все более сложных хитросплетений. В конечном счете весь экран будет заполнен "турбулентным" узором, подобно показанному на рис. 6.3b (четырехкратная симметрия начальной конфигурации, конечно, сохраняется этим инвариантным к повороту правилом неограниченно долго).

Заметим, что на протяжении всего этого процесса контур квадрата не разрушается (фактически он продолжает генерировать волны, идущие как внутрь, так и наружу). Тот же эксперимент, начинающийся с большого круга - (а) на рис. 6.4 - дает узор (Ь) после 4000 шагов. Так как динамика обратима, то если мы вернемся во времени назад на то же число шагов, то возвратимся к узору (а): турбулентность "уничтожена". Давайте проделаем все это вновь, но на этот раз, когда мы находимся в (Ь), до включения заднего хода заменим только один бит плоскости 0 вне круга. Когда мы двинемся назад,

[1] То есть из единиц в обеих плоскостях, как в табл. (6.2).

эта "ошибка" будет быстро расти, разрушая тонкие взаимосвязи, скрытые в (Ь), а итоговым узором "t » 0" станет (с) вместо (а): полный хаос вне круга. Однако внутри круга мы возвратились к правильной исходной конфигурации: возмущение не распространилось на внутренность круга.

Ясно, что граница круга действует как непроницаемый барьер, полностью изолирующий то, что находится внутри, от того, что находится снаружи, и наоборот. При таких начальных условиях система фактически расчленена на две *несвязанные* подсистемы.

Тот факт, что края круга сохраняются, является выражением простого *закона сохранения*. Законы сохранения в клеточных автоматах обсуждаются в [35] и [45].

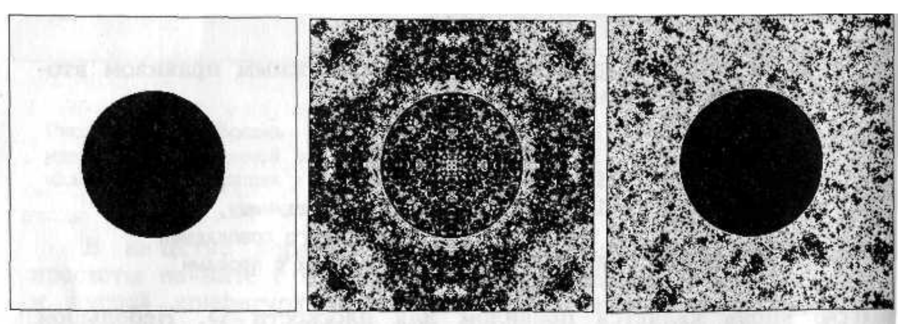


Рис. 6.4. TIME-TUNNEL. От (а) до (Ь) вперед во времени и от (Ь) до (с) назад во времени после замены одного бита в (Ь).

6.4. Другие примерь!

Существует много обратимых правил, тесно связанных с правилом TIME-TUNNEL - примером из предыдущего раздела. Возьмем, например, оператор выбора, который является частью определения TIME-TUNNEL, и заменим его любой другой таблицей, в которой все состояния возвращают то же самое значение, за исключением состояния с максимальным и минимальным числом живых соседей (или обоих этих состояний). Каждое из этих правил приводит к двум причинно несвязанным областям, когда эволюция начинается с конфигурации, показанной на рис. 6.4а. Все обобщения на любую другую симметричную окрестность, в которой все состояния, кроме максимального и минимального числа соседей, рассматриваются аналогично, также обладают этим свойством. Это свойство возникает потому,

что клетки, примыкающие к границе черного и белого на рис. 6.4, никогда не содержат в своих окрестностях максимального или минимального числа соседей - дополнительные пары вдоль границы всегда остаются дополнительными.

Правило PARITY-FLIP также может породить семейство взаимосвязанных обратимых правил. Это семейство включает всякое правило, которое зависит от операции XOR между любым набором соседей в настоящем и центральной клеткой окрестности в прошлом. Каждое из этих правил, будучи линейным, обладает этим свойством наряду с PARITY-FLIP (см. примечание на стр.35).

Вы можете взять любое правило первого порядка и обнаружить тесно связанное с ним обратимое правило второго порядка, используя операцию XOR от результата этого правила и текущего состояния центральной клетки, а также ее прошлого состояния. Последняя операция делает правило обратимым (см. разд. 14.2), тогда как предыдущая гарантирует, что паттерны, которые не изменяются в правиле первого порядка, являются неизменяющимися и в версии второго порядка.

Объем книги не позволяет нам привести развернутое феноменологическое обсуждение обратимых правил второго порядка. Дальнейшие примеры появятся в гл. 9 и третьей части этой книги.

Глава 7

СОСЕДИ И ОКРЕСТНОСТИ

В этой главе мы представляем в систематическом виде источники информации, к которым клетка САМ имеет доступ с целью вычисления своего нового состояния.

Напомним, что САМ производит вычисление посредством просмотра таблицы. Функционально САМ ведет себя так, как если бы каждая клетка обладала своей собственной копией таблицы правил и все клетки заменялись одновременно. В действительности же таблица существует в единственном экземпляре, доступ к которому открыт всем клеткам в порядке очереди; однако конвейерный механизм, необходимый для превращения этого последовательного процесса в эффективно параллельный, скрыт от глаз, и пользователю никогда не придется о нем думать.[1] В частности, известно, что в то время, когда клеточный автомат создает значения для момента $t+l$ (это время клеточного автомата, и оно увеличивается на единицу с каждым шагом), на входах таблицы появляются только значения для момента времени t: новое значение, сконструированное клеткой во время последовательного обновления массива, скрыто от глаз, в то время как ее текущее значение остается видимым, пока не будет заменен весь массив.

Таблица правил порождает новое значение для каждого из битов, которые образуют клетку. Потенциально пригодные источники информации слишком многочисленны, чтобы их можно было одновременно представить в справочной таблице практически приемлемого размера[2]. Важным аспектом в конструировании САМ был выбор функциональных групп переменных, используемых совместно или поочередно в рамках ограничений, устанавливаемых размером таблицы, которые дали бы

[1] Единственный параметр, о котором следует помнить при подключении внешних аппаратных средств к САМ, ЭТО период внутренних часов - около 180 нсек. С каждым импульсом часов на входах таблицы возникает множество значений - состояния соседей; во время следующего импульса часов с выхода таблицы считывается новое значение для клетки, которая должна быть заменена, и в таблицу вводится новое множество входных данных. Таблица могла бы (и действительно может) быть заменена любой комбинационно-логической схемой, которая, получив произвольный набор аргументов, вычисляет определенный результат в пределах периода часов.

[2] Число входов в справочную таблицу никогда не может быть большим, так как размер таблицы экспоненциально зависит от числа входов (удваивается с каждым новым входом); с другой стороны, число сигналов, которые имеются на пульте САМ И ДЛЯ которых предположительно можно найти применение в качестве аргументов таблицы в тот или иной момент, приближается к сотне.

наиболее богатый диапазон возможностей в исследовании динамики клеточного автомата.

Во многих ситуациях правило сможет использовать помимо обычных пространственных соседей дополнительных *псевдососедей*. Эти переменные могут нести информацию, зависящую от пространственных или временных факторов, параметры, сообщаемые компьютером, в который встроен САМ-модуль, и сигналы, обеспечиваемые внешними аппаратными средствами (такими как источник случайных чисел или даже видеокамера).

7.1. Слабо связанная пара

Машина САМ функционально организована в виде идентичных "половин", именуемых САМ-А и САМ-В. Каждая половина состоит из двух плоскостей битов (плоскости 0 и 1 для САМ-А, 2 и 3 для САМ-В) и справочной таблицы и способна выполнить правила клеточного автомата совершенно независимо от другой. Однако каждая из двух частей может видеть кое-что из того, что происходит в других; следовательно, две половины могут быть также запущены как единый клеточный автомат, состоящий из двух связанных подсистем. Эта связь обычно относительно слабая: только соседи CENTER и CENTER' могут восприниматься другой половиной.

Для специальных применений связь между двумя подсистемами может быть усилена путем создания машины с пользовательской конфигурацией. В САМ все относящиеся к делу входные и выходные линии вынесены на *пользовательский коммутатор,* так что различные части внутренних цепей для обновления клеток могут быть дополнены или заменены внешними цепями, которые во многих практически интересных случаях могут состоять всего лишь из нескольких перемычек.

Другой способ связать две части состоит в том, чтобы присоединить их край к краю так, чтобы получить один клеточный автомат большего размера. Этот вид операции кратко обуждается в разд. В.2.

Некоторые из средств пульта САМ совместно используются САМ-А и САМ-В; а именно цветовая карта, которая кодирует совокупное содержание двух половин в единое цветное изображение, и счетчик клеточных событий, информация в который вводится отдельным выходом с цветовой карты. Симметричность двух частей распространяется и на способ использования ими общих ресурсов.

В силу указанных выше причин при программировании САМ удобно в каждый момент сосредоточивать внимание только на одной из двух половин. Вследствие их симметричности одно и то же множество термов будет годиться для любой из

них; например, слово NORTH в CAM-A обращается к биту северного соседа клетки в плоскости 0, а в CAM-B - к ее северному соседу в плоскости 2.

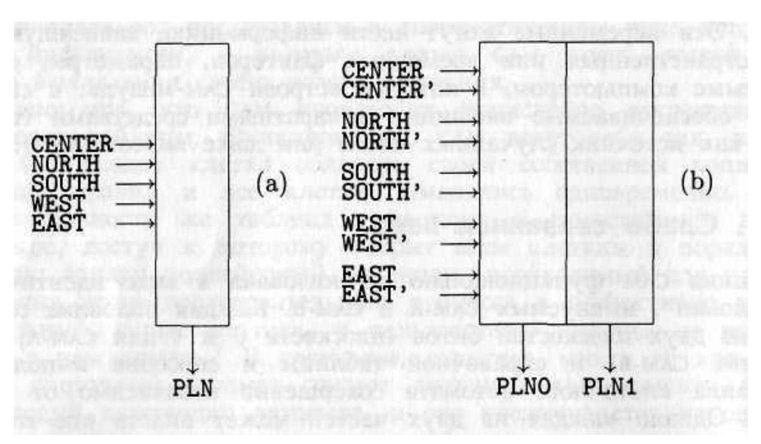


Рис. 7.1. Правило клеточного автомата как черный ящик. Каждый бит состояния окрестности поступает на отдельный вход, а каждый бит нового состояния клетки требует одного выхода.

7.2. Волшебное число двенадцать

В четвертой главе мы рассматривали в качестве иллюстрации один простой класс клеточных автоматов, а именно клеточные автоматы, состоящие из одной плоскости битов (т. е. один бит на клетку) и использующие окрестности из пяти соседей. Для любого автомата этого класса правило можно было закодировать в таблице из 2^5 (=32) элементов. Такую таблицу можно наглядно представить как "черный ящик" (рис. 7.1a), имеющий пять входных и одну выходную линию. Входы представляют биты пятиклеточной окрестности, просматриваемые клеткой, а выход представляет бит, который должен кодировать новое состояние клетки. С другой стороны, если бы клеточный автомат был составлен из двух плоскостей битов 0 и 1 и, следовательно, имел бы по два бита на клетку, то каждая из пяти соседних клеток добавляла бы два бита (один для плоскости 0 и один для плоскости 1), а новое состояние клетки также состояло бы из двух битов; соответствующий черный ящик имел бы 10 входов и 2 выхода, как на рис. 7.1Ь. Сама таблица должна состоять из 2^{10} (=1024) элементов, по два бита на каждый элемент; будет удобно наглядно представлять такую таблицу состоящей из двух *столбцов,* каждый шириной

в один бит и длиной 1024 бит, порождающих выходы, помеченные PLN0 и PLN1.

Справочная таблица, в действительности используемая каждой половиной CAM, имеет ширину четыре столбца, как показано на рис. 7.2. (Два других столбца используются для вспомогательных функций, обсуждаемых позже.) Таблица имеет длину 4096 элементов, соответствующую двенадцати входным (или *адресным)* линиям; это число 12 будет основным ограничением в последующих обсуждениях, так как оно устанавливает верхнюю границу количества информации, которая может быть непосредственно использована таблицей при вычислении нового состояния клетки.

Попробуем наглядно представить смысл этого ограничения. Внутри каждой клетки CAM-A мы имеем идентичную копию черного ящика, аналогичного изображенному на рис. 7.2, с четырьмя выходными портами и двенадцатью кабелями, свисающими с его входного разъема. Предположим, что каждый входной кабель заканчивается контактом ("щупом"), который может быть присоединен к любой переменной, значение которой мы хотим учесть. Скажем, мы хотим запустить демонстрационный пример LIFE, как в гл. 3. Мы присоединяем первые девять контактов к девяти соседям клетки, считывающей биты только с плоскости 0, а именно CENTER, NORTH, ..., S.EAST. (Так как и в CAM-A, и в CAM-в существует только один черный ящик, которым по очереди пользуются все клетки, то по нашему сценарию мы должны предположить, что все, что мы делаем с контактами одной клетки, имитируется всеми клетками массива.) В предположении, что столбец PLN0 таблицы внутри ящика надлежащим образом заполнен, этого достаточно для того, чтобы запустить LIFE без ухищрений; все, что поступает с первого выходного порта, должно становиться новым состоянием клетки - опять же, лишь настолько, насколько это касается плоскости 0.

Если мы хотим ввести свойство ECHO, то соответственно заполняем также столбец PLN1 таблицы, а все, что поступает со второго выходного порта, становится новым состоянием бита в плоскости 1; мы не должны использовать большего числа зажимов, так как все, что необходимо для ECHO, так это знание бита клетки CENTER, который мы уже связали с самым первым контактом. Три контакта все еще не используются.

Теперь мы хотим ввести свойство TRACE. Так как мы хотим считывать не только бит центральной клетки в плоскости 0. но и в плоскости 1 (а именно CENTER'), то должны присоединить еще один контакт. Остались только два контакта! К тому же вспомним, что в плоскости 1 мы видим только один из девяти битов, содержащихся в окне 3x3 с центром в данной клетке. При наличии только 12 контактов мы никогда не

сможем увидеть одновременно девять битов из плоскости 0 и девять битов из плоскости 1.

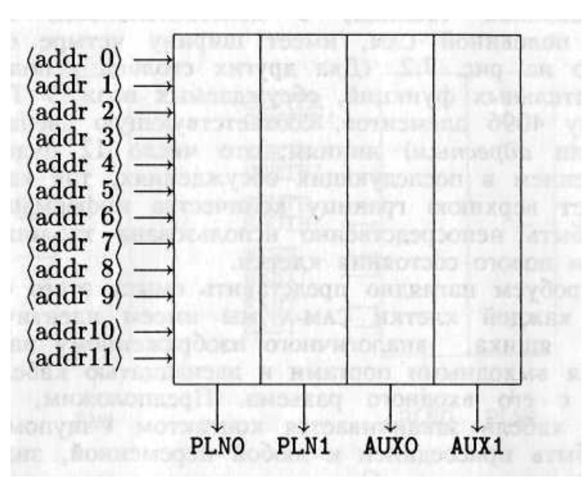


Рис. 7.2. Справочная таблица САМ-А. (Для таблицы САМ-В замените 0 и 1 в метках выходов на 2 и 3. Двенадцать входных "клемм" можно соединить со множеством сигналов.)

Но существуют и другие параметры, которые мы могли бы захотеть узнать: несколько битов с двух других плоскостей (из другой половины машины), некоторая информация о пространственном расположении (можем ли мы создать клеточный автомат типа "шахматной доски", в котором черные и белые квадраты подчиняются разным правилам?), некоторая зависящая от времени информация, предоставляемая компьютером-хозяином, внешние входы, источники случайных чисел и т. д. В определенных случаях можно будет целенаправленно комбинировать информацию, приходящую от различных источников, и направлять ее на один контакт.

Вычисление по справочной таблице осуществляется быстро, но требует тщательного распределения наиболее важных ресурсов, а именно адресных линий таблицы. В ранних прототипах "контакты" были реализованы буквально: как провода, которые можно было подсоединять в любом порядке вручную. Дополнительные схемы могли быть вставлены где угодно. Возвращение к машине, после того как кто-нибудь другой поработал на ней, всегда было сюрпризом! В официальной версии САМ этот подход "сделай сам" по-прежнему возможен, хотя и в чуть более дисциплинированном виде, посредством доступа к пользовательскому коммутатору. Тем не менее важно уметь создавать конфигурацию машины из ряда стандартных "форм",

пригодных для широкого диапазона применений. Они должны быть вызываемы из программ, хорошо документированы и всегда доступны - мы хотим, чтобы наши эксперименты могли сообщаться другим и воспроизводиться.

Именно этим задачам служат "окрестности" CAM.

7.3. Объявление окрестностей

Следующие соображения применяются независимо к каждой половине CAM, т. е. как к CAM-A, так и к CAM-B.

В CAM *сосед* определяется как любой однобитовый источник информации, к которому присоединяется один из 12 контактов или адресных линий справочной таблицы. *Окрестность* представляет собой назначение некоторых или всех этих линий определенным соседям. В общем случае мы будем делать *основное* назначение, которое явно присоединяет *десять* из этих источников к специальному множеству источников сигналов, при помощи объявления вида

м/<основное назначенио

(где N/ является мнемоникой для слова "neighborhood" (окрестность)); источники для оставшихся двух линий определяются факультативным дополнительным назначением вида

&/<дополнительное назначенио.

Новое основное назначение заменяет предыдущее основное назначение и обнуляет предыдущее дополнительное назначение.

Мы можем создать разные варианты окрестностей для двух частей CAM. Слова CAM-A И CAM-B используются для указания, что последующие назначения окрестности должны передаваться только в соответствующую часть машины. CAM-AB направляет назначения в обе части - этот вариант эффективен в начале нового эксперимента.

Пользователю не нужно отслеживать, куда присоединены двенадцать адресных линий, так как объявление каждой окрестности назначает ей удобное символическое имя.[1]

7.3.1. Основные назначения

Объявление *окрестности Мура,* N/MOORE, делает следующие десять слов соседства доступными для использования при определении правила в CAM Forth:

Для справки список этих назначений имеется в таблицах (7.2) и (7.3).

CENTER NORTH SOUTH WEST EAST
N.WEST N.EAST S.WEST S.EAST
CENTER'

В то же время оно присоединяет соответствующие физические сигналы к первым десяти входам в справочную таблицу.[1] Это окрестность, которую мы использовали в предыдущих главах.

Если явно не оговорено противное, все слова соседства в CAM Forth представляют бинарные переменные со значениями О и 1.

Окрестность *фон Неймана,* N/VONN, также задает десять источников сигналов, но подборка в этом случае следующая:

CENTER NORTH SOUTH WEST EAST
CENTER' NORTH' SOUTH' WEST' EAST'

Таким образом становится доступным все содержимое пяти клеток, и в рамках этого подхода можно с полной общностью программировать клеточный автомат, имеющий четыре состояния на клетку.

Особенно удобно в этом контексте бывает оперировать пятью переменными, принимающими четыре возможных значения, т. е. переменными

CENTERS NORTHS SOUTHS WESTS EASTS ,

а не десятью двузначными переменными. В CAM Forth всякий раз, когда в заданной окрестности доступны два варианта слова соседства, со штрихом и без штриха, доступна также и "объединенная" версия (заканчивающаяся на 's'). Например, CENTERS определяется как CENTER + 2xCENTER' и принимает значения 0, 1, 2 и 3.

Выделенная пара CENTER, CENTER' (и, следовательно, CENTERS) включается в каждое основное назначение.

Третье основное назначение, именуемое окрестностью *Марголуса,* будет представлено в гл. 12. Оно появляется в трех разновидностях, именуемых N/MARG, N/MARG-PH и N/MARG-HV.

Как мы уже отметили, приведенные выше объявления являются взаимоисключающими: любое из них отменяет предыдущее. Но где эти десять контактов располагаются *до* назначения окрестности? В действительности существует четвертый вариант - по умолчанию, именуемый *пользовательской* окрестностью, которая автоматически выбирается в на-

[1] Сигналы и таблица либо из CAM-A, либо из CAM-B, В зависимости от ситуации.

чале нового эксперимента и к которой можно возвратиться явным объявлением N/USER. Ее источниками сигнала являются

```
CENTER
CENTER'
```

•пользователь 2> <пользователь 3> <пользователь 4>
<пользователь 5> <пользователь 6> <пользователь 7>
<пользователь 8> <пользователь 9>

Последние восемь соседей соответствуют штеккерам на пользовательском коммутаторе и могут быть присоединены к произвольным источникам сигналов из внешнего мира. Заметим, что мы не назначали имен этим источникам, так как пользователю будет удобнее дать те мнемоники, которые он хочет, и соответствующие тому, что действительно присоединено к ним. Для того чтобы присоединить, скажем, имя соседа "CAMERA" к адресной линии <addr 7> (см. рис. 7.2), которая в существующем назначении соседей прямо присоединена к внешнему входному сигналу <пользователь 7>, пишем просто

```
7 == CAMERA .                    (7.1)
```

Конечно, существующие слова соседства могут быть в любой момент переименованы, чтобы удовлетворить чей-то вкус. Например, всегда допустимо написать

```
: UNDERLAY
CENTER' ;
```

и затем обращаться к UNDERLAY, а не к CENTER', в теле правила.

7.3.2. Дополнительные назначения

После того как мы сделали основные покупки, у нас осталось немного мелочи, а именно две адресные линии, которые мы, возможно, захотим потратить на некоторые второстепенные приобретения. В конце концов станет ясно, что эти приобретения могут дать нам большую власть.

Дополнительное назначение &/CENTERS присоединяет последние два контакта к другой половине машины, предоставляя нам окно 1x1 в двух других плоскостях; эти дополнительные соседи называются

```
&CENTER
&CENTER'
```

или, в совокупности, &CENTERS.

Общим и очень мощным методом расширения диапазона поведения, который может быть исследован в машине клеточных автоматов, является *композиция правил.* То есть, совершая один шаг по правилу *a,* один шаг по правилу *b* и так далее в определенной последовательности, мы можем эффективно сконструировать "суперправило" со свойствами, недоступными его отдельным составляющим.

Загрузка нового правила с компьютера-хозяина в CAM перед каждым шагом требует некоторого времени[1]. Во многих случаях мы сможем сразу сделать предварительную загрузку справочной таблицы с несколькими правилами (они будут занимать различные части таблицы), а уж затем выполнить их в циклической последовательности без прерываний. Мы можем представить всю таблицу как единое суперправило, которое переключается от *a* к *b* и так далее на последовательных шагах. Зависящая от времени информация, необходимая для выполнения этих переключений, предоставляется псевдососедями, которые могут быть обработаны компьютером-хозяином между шагами (псевдососеди считываются правилом точно так же, как если бы они были обычные пространственные соседи).

Объявление &/PHASES присоединяет два дополнительных контакта к двум из этих псевдососедей, а именно

&PHASE
&PHASE'

известных также CAM Forth под общим именем &PHASES. (Заметим, что &PHASE и &PKASE следуют обычной схеме именования со штрихом/без штриха для пар связанных переменных; однако они сами по себе не связаны с определенными плоскостями. Заметим также, что встроенные псевдососеди CAM, такие как &PHASE и &PHASE, а также &HORZ и &VERT, введенные ниже, существуют в единственном экземпляре, совместно используемом CAM-A И CAM-B.)

Наконец, младшее назначение &HV выбирает двух псевдососедей, именуемых

&HORZ
&VERT

{горизонтальная фаза и *вертикальная фаза),* которые дают некоторую пространственную информацию; в частности, они

[1] В типичных случаях это время одного шага эволюции для предварительно скомпилированного правила, которое было запомнено как таблица в памяти компьютера-хозяина; много больше, если правило должно быть сначала скомпилировано.

позволяют сформулировать правила, которых следуют конфигурациям "в полоску" или "в клетку", и поддерживают метод разбиения, обсужденный в гл. 12. Они будут объяснены более детально ниже, в разд. 11.1. Общее имя для этой пары псевдососедей - &HV (=*HORZ + 2xVERT).

Дополнительные назначения также являются взаимоисключающими. Назначением по умолчанию является &/USER, с помощью которого мы можем дополнить основной выбор парой внешних линий; т. е. этот выбор вводит двух (неименованных) соседей

<пользователь
•пользователь 11> ,

связанных с двумя дополнительными штеккерами на пользовательском коммутаторе.

7.4. Сводка окрестностей

Мы вновь подчеркиваем, что количество соседей в любой момент равно количеству адресных линий справочной таблицы, как показано на рис. 7.2.

Конструкция '<адресная линия>==<имя соседа>', используемая выше, фактически используется программным обеспечением CAM для порождения имен соседей, связанных с различными окрестностями. В этой форме записи полный список основных окрестностей и соответствующих им назначений может быть представлен в следующем виде:

addr	N/M00RE	N/VONN	N/MARG	N/MARG-PH	N/MARG-HV	N/USER
0 ==	CENTER	CENTER	CENTER	CENTER	CENTER	CENTER
1 ==	CENTER"	CENTER'	CENTER'	CENTER'	CENTER'	CENTER'
2 ==	S.EAST	EAST'	CW	CW	CW	(user 2)
3 ==	S.WEST	WEST'	CCW	CCW	CCW	(user 3}
4 ==	N.EAST	SOUTH'	OPP	OPP	OPP	(user 4}
5 ==	N.WEST	NORTH'	CW	CW	CW	(user 5)
6 ==	EAST	EAST	CCW'	CCW'	CCW'	(user 6)
7 ==	WEST	WEST	OPP'	OPP'	OPP'	(user .7)
8 ==	SOUTH	SOUTH	(user 8)	PHASE	HORZ	(user 8)
9 ==	NORTH	NORTH	(user 9)	PHASE'	VERT	(user 9}

(7.2)

(для окрестностей Марголуса соседи будут подробно описаны в гл. 12). Кроме того, каждая окрестность делает доступной (в

качестве дополнительного удобства) соответствующую объединенную версию для каждой пары соседей, отличающихся наличием штриха (например, CENTERS для CENTER и CENTER'), а также HV для пары HORZ-VERT.

Подобным образом, дополнительные окрестности предоставляют варианты

addr	&/CENTERS	&/PHASES	&/HV	&/USER
10 ==	&CENTER	&PHASE	&HORZ	(user 10)
11 ==	&CENTER'	&PHASE'	&VERT	(user 11)

(7.3)

с аналогичными средствами для объединенных пар соседей.

7.5. Заказные окрестности

Вам может понадобиться окрестность, которая использует смесь сигналов, отличную от предоставляемых стандартными окрестностями, список которых приведен выше, или вы, возможно, захотите использовать некоторые сигналы, доступные на пользовательском коммутаторе, но совершенно не связанные ни с одной из стандартных окрестностей. В таких случаях легко превратить пользовательскую окрестность (основную, дополнительную или обе) в *заказную* подключением нескольких перемычек в пользовательском коммутаторе. Эта процедура объясняется в разд. 9.7.

7.6. Создание таблиц

Как мы увидели, CAM содержит несколько таблиц, а именно цветовую карту и две справочные таблицы, связанные с CAM-A и CAM-B. Все эти таблицы могут использоваться в сложном эксперименте, а их содержание может изменяться по ходу эксперимента. Важно иметь общую процедуру для определения желаемого содержимого соответствующих столбцов этих таблиц.

Команда MAKE-TABLE <дескриптор таблицы> заполняет один или более столбцов таблицы так, как указывает <дескриптор таблицы>, определенный пользователем посредством слова языка Forth, содержащего необходимые определения.

"Правила", которые мы использовали в предыдущих главах, такие как

```
: ECHO
CENTER >PLN1 ;
```

являются типичными примерами подобных дескрипторов; фактически команда

```
MAKE-TABLE ECHO
```

перезаписывает столбец PLN1 таблицы ECHO, оставляя другие столбцы без изменений (см. разд. 4.2).

За один раз может быть заполнено и несколько столбцов. Например, последовательность

```
СДМ-A N/VONN
                                        : PARITY-WITH-ECHO
   CENTER NORTH SOUTH WEST EAST
          XOR XOR XOR XOR >PLNO
                    CENTER >PLN1 ;
MAKE-TABLE PARITY-WITH-ECHO
```

вписала бы в столбцы PLNO и PLN1 информацию, требуемую для того, чтобы выполнить правило PARITY (см. разд. 4.2) с включенным свойством ECHO.

Эта двухстадийная процедура - сначала определяется дескриптор, а затем он передается в качестве аргумента в команду - дает необходимую в сложных ситуациях гибкость.

Для того чтобы иметь больше возможностей управления, можно было бы определить PARITY как в разд. 4.2, а затем подготовить следующие дескрипторы:

```
                          : ECHO
            CENTER >PLN1 ;
                          : TRACE
    CENTER CENTER' OR >PLN1 ;
                          : BARE
                 0 >PLN1 ;
```

Тогда, начав эксперимент с

```
MAKE-TABLE PARITY
MAKE-TABLE BARE
```

можно было бы включать и выключать режимы ECHO или TRACE по желанию подачей соответствующих команд MAKE-TABLE.

Заметим, что при обычных обстоятельствах не нужно будет вводить эти команды с клавиатуры посимвольно в процессе моделирования; CAM Forth предоставляет средства для "закрепления" произвольной команды или последовательности команд за отдельной клавишей, преобразуя, таким образом, клавиатуру в пульт управления для выполнения эксперимента в реальном времени.

В приведенных выше примерах было естественно отдельно определять содержимое столбцов таблицы PLNO и PLN1, так как там две плоскости битов играли совершенно независимые роли.

В других ситуациях удобнее оказывается возвращать новое состояние клетки как одну переменную с четырьмя возможными значениями. Слово >PLNA принимает такую переменную в качестве аргумента и записывает ее как двухбитовой элемент таблицы, т. е. поперек столбцов PLN0 и PLN1. Например, чтобы заставить клеточный автомат пройти через состояния о и з, необходимо написать

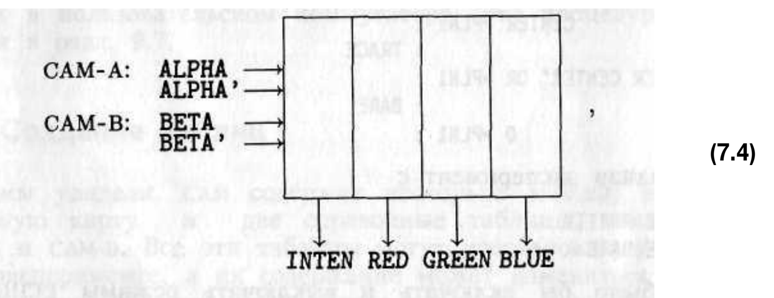
```
                          : 4CYCLE
    CENTERS  1+  4 MOD  >PLNA ;
MAKE-TABLE  4CYCLE
```

Последовательность 4 MOD делит аргумент на стеке на 4 и возвращает остаток, гарантируя, таким образом, что после достижения состояния 3 клетка возвратится в состояние О.[1]

Соответствующими словами ("диспетчерами столбцов") для САМ-В, конечно, являются >PLN2, >PLN3 И >PLNB. Подобные слова, такие как >AUX0, >AUXA и т. д., используются для заполнения вспомогательных столбцов справочной таблицы (см. разд. 7.2).

7.7. Цветовая карта и счетчик событий

Цветовая карта, вкратце введенная в разд. 2.2.2, имеет структуру, аналогичную структуре справочных таблиц, а именно



(7.4)

а ее содержимое определяется аналогично заданием дескриптора таблицы как аргумента команды MAKE-CMAP. Столбцы, помеченные INTEN, RED, GREEN и BLUE, заполняются словами-"диспетчерами" >INTEN, >RED и т. Д.

Две пары входов ALPHA, ALPHA' и BETA, BETA' подходят соответственно от CAM-A и САМ-В. Обычно они присоединены к плоскостям от нулевой до третьей включительно и, таким об-

[1] **Так как Forth кодирует целые как 16-разрядные двоичные числа, а >PLNA только передает в таблицу два наименьших значащих бита его аргумента (см. разд. А. 14), то конструкция 4 MOD является в этом случае избыточной.**

разом, непосредственно видят четыре бита, которые образуют текущее состояние клетки. Второй вариант состоит в том, чтобы присоединить их к вспомогательным выходам справочной таблицы от AUX0 до AUX3 включительно; таким образом, возможно послать на цветовую карту произвольную *функцию всей окрестности клетки.* Мы будем считать, что эти два варианта выбираются командами SHOW-STATE (выбор по умолчанию) и SHOW-FUNCTION.

Например, цветовая карта таблицы (3.1) определяется следующим образом:

```
                         : ECHO-MAP
              0   >INTEN
ALPHA   ALPHA'     AND >RED       \ 1 в обоих плоскостях
ALPHA   ALPHA' NOT AND >GREEN     \ 1 только в плоскости 0
ALPHA NOT ALPHA'   AND >BLUE    ; \ 1 только в плоскости 1

MAKE-CHAP ECHO-MAP
```

Другие примеры можно найти в разд. 9.2 и 12.8.2.

Саму по себе или в сочетании со вспомогательными таблицами цветовую карту можно заставить выполнять - одновременно с собственно моделированием - существенный объем препроцессорной обработки содержимого массива до пересылки его на монитор для *визуального* анализа в реальном времени. Зачастую, однако, необходимы "зарегистрированные" данные, предназначенные для дальнейшего параллельного или последовательного компьютерного анализа. Для этой цели выход "интенсивность" также передается в счетчик событий, который на каждом шаге регистрирует число пикселей повышенной интенсивности; значение этого счетчика непосредственно доступно компьютеру-хозяину.

Надлежащим программированием входа счетчика можно обнаружить и сосчитать определенные явления локальной природы, сравнить содержимое плоскостей, вычислить корреляции "на лету", набрать статистику, а также автоматически остановить моделирование при появлении определенных условий. Различные виды событий могут быть классифицированы с помощью многократных прогонов со счетчиком.[1] Хотя разрешающая способность счетчика и ограничивается обычными требованиями *локальности и однородности,* он может быть использован для того, чтобы существенно уменьшить число ситуаций, в которых требуется компьютер-хозяин для приостановки моделирования, считывания всего или части содержимого плоскостей битов и извлечения соответствующей информации.

[1] **Это, конечно, требует "разметки времени" для одного или большего числа шагов.**

Глава 8

СЛУЧАЙНОСТЬ И ВЕРОЯТНОСТНЫЕ ПРАВИЛА

Правила, описанные в предыдущих главах, являются *детерминистскими,* т. е. новое состояние клетки однозначно определяется текущим состоянием ее соседей: отправляясь от тех же начальных условий, неизбежно получаем ту же эволюцию.

В случае *вероятностного* правила одна и та же текущая ситуация может привести к нескольким различным результатам с заданной вероятностью каждого из них. Скажем, мы просматриваем определенный вариант таблицы с целью определить новое состояние клетки и вместо одного значения мы находим два, *a* и *Ь,* и сообщение: "Чтобы сделать окончательный выбор, киньте монету!". Если монета настоящая, то с вероятностью одна вторая будет выбрано значение *a.* Монета, у которой одна сторона тяжелее другой, даст иную вероятность и, следовательно, другое правило: вращением ручки управления источника случайности, если бы она была, из одной и той же справочной таблицы можно было бы получить все множество вероятностных правил, покрывающих непрерывный диапазон, ограниченный двумя детерминированными элементами "всегда а" и "всегда *Ь*".

Вероятностные правила полезны для многих задач моделирования.

В САМ каждый элемент справочной таблицы содержит единственное и точно определенное значение. Тем не менее можно легко синтезировать недетерминированные исходы. Например, можно связать один из входов таблицы (см. гл. 7) со случайной двоичной переменной: для заданного присвоения значений всем другим входам, выходное значение будет поступать от того или иного из двух разных элементов таблицы в зависимости от текущего значения случайной переменной. Таким образом, справочная таблица будет возвращать вероятностные результаты с распределением вероятности, непосредственно связанным с распределением случайной переменной.

Этот подход можно обобщить. Мы будем использовать термин *шумящий сосед* для некоторой величины, которая может играть роль случайной переменной в определении правила. Шумящие соседи могут быть заданы внешними аппаратными средствами или же сгенерированы внутри САМ с помощью разнообразных методов.

8.1. Экспоненциальное затухание

Рассмотрим большое число зажженных свечей, помещенных на дождь. Как только капля дождя попадает на свечу, та гаснет. Чем меньше остается свечей, тем меньше гашений будет наблюдаться в следующий момент: все освещение будет затухать экспоненциально.

Мы будем использовать плоскость 0 для представления массива свечей (1="зажжена", о="погашена"). Мы хотим, чтобы на каждом шаге капли гасили некоторые свечи: сколько и каких? Другими словами,

> Как нам создать случайный паттерн капель дождя?
>
> Как нам заставить воздействовать его на массив свечей?
>
> Как нам создать после каждого шага новый случайный паттерн?

Мы опишем сначала непрактичное, но очень простое решение. Плоскость 1 будет использоваться для представления капель дождя (1="капля", о="нет капли"). Перед каждым шагом мы заполняем плоскость 1 новым случайным паттерном *вручную* и уж затем выполняем шаг: каждая капля дождя попадает на соответствующую свечу (если та все еще зажжена). Дождевые капли действуют на свечи согласно следующему правилу DECAY (для удобства мы переименовали CENTER'):

```
              : RAND
     CENTER' ;
              : DECAY
RAND { CENTER 0 } >PLNO ;
```

То есть, если капли нет (RAND=O), свеча останется в своем текущем состоянии; однако капля дождя (RAND=1) погасит свечу.

Мы выполним эксперимент с цветовой картой, который сделает содержимое плоскости 1 (случайный источник) невидимым и изобразим только содержимое плоскости 0 (свечи). Эксперимент начнем при всех зажженных свечах. Давайте заполним плоскость 1 случайным узором, имеющим определенную плотность. Мы можем, например, пять раз кинуть монету и если получим все пять раз "орел", то поместим каплю в первую клетку массива; это произойдет с вероятностью $p=1/2^5$. Мы повторим это для второй клетки и т. д., до тех пор пока каждая из N (=256x256) клеток не получит возможность быть заполненной: около pN (=2048) клеток будут содержать каплю. Когда узор завершен, сделаем один шаг пра-

вила DECAY. Изрядное число, свечей будет погашено. Повторим всю процедуру и погаснут еще несколько свечей. Таким образом, весь массив свечей будет постепенно погашен в соответствии с нужным нам экспоненциальным законом затухания, как показано на рис. 8.1.

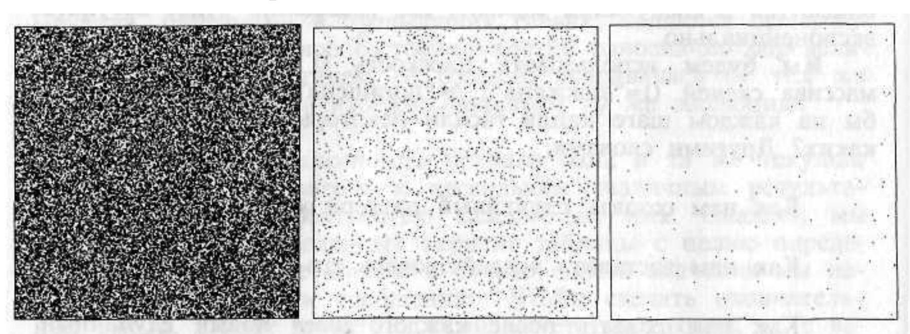


Рис. 8.1. Экспоненциальное затухание, управляемое случайным источником.

8.2. Простой генератор шума

Создание случайного паттерна вручную исключительно медленное занятие. Чтобы ускорить дело, мы можем попросить головной компьютер порождать случайные биты с вероятностью *p* и вставлять их в массив, но даже это медленно и требует нескольких секунд для создания экрана, заполненного случайными битами. В действительности нам нужно нечто, способное производить случайные биты настолько быстро, насколько CAM может их использовать, т. е. порядка шести миллионов операций в секунду.

Легко построить внешнюю схему, которая будет производить новый случайный бит для каждого импульса часов CAM.[1] Мы можем подключить выход этого, источника, скажем, к штеккеру <пользователь 11> CAM. Как мы объяснили в гл. 7, определение

И == RAND

создаст имя RAND для этого псевдососеда, а назначение

[1] Регистр сдвига умеренной длины (скажем, 31 бит) с линейной обратной связью даст "синтетическую монету", которая для большинства практических целей неотличима от теоретической подлинной монеты [44]. Если мы хотим, чтобы единицы порождались с вероятностью, которая регулируется небольшими приращениями на всем отрезке от нуля до единицы, а не с фиксированной вероятностью 1/2, то необходимы более сложные схемы.

```
&/USER
```

направит физический сигнал пользователь 11> в адресную линию 11 справочной таблицы.

Обратившись к внешним аппаратным средствам, можно получить случайный источник, аппроксимирующий желаемые статистические свойства. Но если CAM хороша для моделирования огромного многообразия систем, то почему бы не попытаться использовать ее и для этой цели, что позволило бы легко ставить и воспроизводить эксперименты на стандартном оборудовании? Действительно, хотя для некоторых критических приложений может быть абсолютно необходим внешний генератор случайных чисел, подход на основе внутреннего порождения более чем адекватен для большинства вероятностных моделей, описанных в этой книге.

В эксперименте с экспоненциальным затуханием, рассмотренном в разд. 8.1, плоскость 1 использовалась как пассивное хранилище для созданного вручную паттерна. Здесь мы преобразуем эту плоскость в непрерывно помешиваемый "бульон битов", содержащий на каждом шаге новый случайный паттерн. Подходящим правилом (использующим окрестность NVONN) является

```
                                        : STIR
   CENTER' NORTH' WEST' SOUTH' EAST'
               AND XOR XOR XOR >PLN1 ;
```

Для соседей, расположенных в указанном порядке, операции XOR гарантируют, что после долгого промежутка времени бульон даст равномерную смесь нулей и единиц (см. примечание на стр.); за счет включения в правило некоторой нелинейности операция AND гарантирует, что система не зациклится в короткопериодных осцилляциях.

Этот бульон будет сразу же готов к употреблению, если мы начнем его приготовление со случайного паттерна, содержащего 50% единиц; однако пригодный к употреблению бульон получится из практически любого начального паттерна после перемешивания в течение нескольких сотен шагов.[1] Статистические свойства этого генератора шума вполне достаточны для нашей цели. Мы могли бы использовать непосредственно необработанные биты плоскости 1 как капли, но это дало бы очень сильный дождь ($p=1/2$). Чтобы получить более низкое значение для *p*, мы введем "фильтр" шума, который возвращает единицу (каплю), только если несколько прилегающих битов бульона находятся в состоянии единица:

[1] **Это время, которое требуется сигналу, чтобы обойти "весь мир" (экран CAM размером 256x256) несколько раз.**

```
                              : RAND (-- 0|1)
  CENTER' NORTH' SOUTH' WEST' EAST'
                  AND AND AND AND ;
```

В этой версии RAND, которая заменит версию предыдущего раздела, вероятность нахождения капли в любом месте $p=1/2^5$, т. е. точно та же, как если бы мы пять раз бросали монету.

Что касается плоскости 0, то слово RAND действует как псевдососед, значение которого случайно меняется от клетки к клетке и от шага к шагу согласно точно определенному распределению вероятности. Возможными значениями шумящего соседа являются 1 и о; вероятность *p* может быть грубо отрегулирована (в степенях двойки) изменением числа операндов, к которым применяется операция AND правила RAND.

Мы позже обсудим, как получить более широкий набор значений, более широкий набор вероятностей и более тонкое разрешение в регулировке вероятности, а также как избежать корреляций.

8.3. Снова правила голосования

Генератор случайных чисел может быть использован для включения некоторого "теплового шума" в другое детерминированное правило.

В разд. 5.4 мы рассмотрели правило голосования 9MAJ, которое за несколько шагов превращает случайную начальную конфигурацию в области, состоящие только из нулей, и области, состоящие только из единиц, разделенные совершенно нерегулярными границами, - замороженный образец локальных альянсов. Чтобы поддержать рост более компактных областей, мы слегка изменили правило: новое правило ANNEAL взбалтывает те области, где большинство, а именно границы между областями, является слабо выраженным и заставляет систему исследовать жизнеспособность более долгоживущих альянсов.

Здесь мы поступаем аналогичным образом, используя на этот раз шум, а не разрушение, для сохранения подвижности на границах. Генератор случайных чисел будет тот же, что и в предыдущем разделе, т. е. шум порождается правилом STIR в плоскости 1 и фильтруется правилом RAND.

Мы начнем с правила большинства, которое использует только пять соседей, т. е.

```
                           : 5SUM (-- n)
  CENTER NORTH SOUTH WEST EAST
                      + + + +  ;
                           : 5MAJ
     5SUM { 0 0 0 1 1 1 } >PLNO ;
```

Это порождает границы областей, аналогичные тем, которые задаются правилом 9MAJ, хотя и чуть более изрезанные (рис. 8.2a).

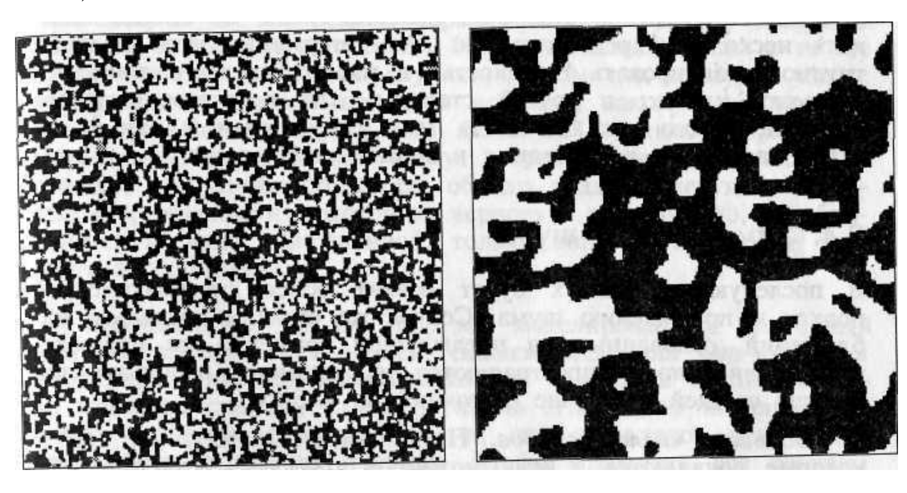


Рис. 8.2 Замороженные границы (a), порожденные правилом 5MAJ, тают в (Ь) под влиянием теплового шума, обеспечиваемого вероятностным вариантом этого правила, а именно RAND-ANNEAL.

В правиле 5MAJ два средних элемента (...0 1...), расположенные чуть ниже и чуть выше 50%-ного порога, соответствуют "пограничным" ситуациям, где никакая партия не обладает подавляющим преимуществом; мы собираемся заменить их вероятностными исходами: о ниже порога будет заменен правилом RAND предыдущего раздела, которое чаще всего возвращает о, но иногда возвращает и 1 (с вероятностью p=1/32); аналогично единица выше порога будет заменена дополнением RAND, т. е. правилом -RAND, которое чаще всего возвращает 1, но иногда возвращает и о:

```
                    : RAND (-- 0|1)
          RAND I XOR ;
```

Итоговое правило, проиллюстрированное на рис. 8.2Ь, это

```
                          : RAND-ANNEAL
    5SUM { 0 0 RAND -RAND 1 1 }
                      >PLNO ;
```

Тот факт, что две случайные переменные RAND и -RAND строго коррелированы (одна является дополнением другой), может нас не беспокоить, так как для любой конкретной клетки мы собираемся использовать только одну или другую и никогда *обе*.

Если увеличить значение *p,* то скорость нормализации будет возрастать, пока в конце концов при *p,* равном единице, правило не перейдет в версию ANNEAL с пятью соседями.

Заметим, что на рис. 5.3Ь правило ANNEAL не смогло сгладить нескольких редких пятен: для детерминистского правила трудно гарантировать "гладкость" в масштабах всего лишь нескольких клеток; с другой стороны, для недетерминистского правила, такого как RANDANNEAL (с *p,* не равным 0 или 1), и такие застывшие образования в конце концов тают.

8.4. Замечания о шуме

В последующих главах будет развит более систематический подход к порождению шума. Сейчас же сделаем несколько наблюдений, основанных на предыдущем примере; эти наблюдения тривиально распространяются на правила, использующие больше соседей и больше состояний на клетку.

Большее число исходов. Пять соседей CENTER', ..., EAST', которые появляются в RAND, можно использовать для порождения более двух различных значений. Например, величина

CENTER' + 2XNORTH'

принимает значения о, 1, 2 и з и, следовательно, ее можно использовать для моделирования бросания четырехсторонней кости. Ясно, что, наблюдая за всеми пятью соседями, можно разрешить вплоть до 32 различных исходов. (При использовании девяти соседей можно получить 512 исходов.)

Градуировка вероятности. Все эти тридцать два исхода имеют равные вероятности.[1] Версия правила RAND, данная в разд. 8.2, использует только один из них и, значит, дает вероятность p=1/32. Можно сконструировать более сложный генератор случайных исходов: создав слово Forth, которое возвращает единицу для *n* исходов (выбранных раз и навсегда) и О для оставшихся 32-га, получаем вероятность *p=n/32.* Таким образом, за счет изменения *n* вероятность *p* можно выбрать в диапазоне от нуля до единицы равной одному из тридцати трех равноотстоящих фиксированных значений (513 значений, если используется девять соседей).

[1] Доказательство этого факта довольно сложно. Вкратце, если рассматривается бесконечно расширяемый клеточный автомат, то наличие двух операций XOR в качестве двух последних операций правила STIR гарантирует, что глобальное отображение, осуществляемое клеточным автоматом, *сюръективно* и что поэтому все его итерации *сбалансированы* (см. [24, 39]). Как следствие, почти все начальные конфигурации (т.е. все конфигурации, за исключением, возможно, подмножества меры нуль) дают устойчивое распределение, где все паттерны конечных размеров представлены с одинаковым весом.

Взаимозаменяемость. В предыдущих разделах мы рассмотрели три различных случайных источника, а именно (а) данные, записанные прямо в плоскости 1; (Ь) случайный внешний источник; и (с) внутреннее перемешивание и взятие операции AND от битов в плоскости *P'*. Мы использовали одно и то же слово RAND во всех трех случаях, чтобы подчеркнуть, что с точки зрения программирования неважно, какой метод используется: различные версии RAND являются функционально взаимозаменяемыми. Обычно пишут вероятностное правило как правило RAND шумящего соседа общего вида, исследуют его поведение, используя некоторую черновую версию RAND, и обращаются к более совершенной, только когда того требует точность эксперимента.

Модульность. До сих пор мы моделировали и клеточный автомат, который нам нужен, скажем DECAY или RANDANNEAL, и связанный с ним случайный источник только на одной половине машины, скажем, CAM-A. Правила STIR и RAND использовали около половины ресурсов[1] CAM-A; это оставляло мало места для работы основного правила. Чтобы сохранить все ресурсы CAM-A для основного правила, на практике рекомендуется реализация случайного источника в CAM-B; это способствует также более структурированному подходу к планированию эксперимента. Если необходимы еще более сложные источники случайности, то в качестве специализированного источника шума можно использовать вторую машину CAM.

Устранение корреляций. Так как правило клеточного автомата локально, то информация может только медленно распространяться из некоторого места. Корреляции, вызываемые конечностью скорости света, можно практически устранить с помощью метода, который использует два модуля CAM и случайным образом смещает точки, с которых в каждом из них начинается обновление, как объясняется в гл. 15.6.

Тонкое разрешение. До сих пор мы рассматривали перемешивающие правила, которые в конечном счете дают равную пропорцию нулей и единиц в плоскости битов, используемой как генератор случайных чисел, даже если исходный зародыш содержит их в другой пропорции. В гл. 12 мы введем правила, которые *сохраняют* число единиц в плоскости битов (они рассматриваются как неразрушимые частицы), несмотря на интенсивную рандомизацию. Таким образом можно точно отрегулировать вероятность *p* (с минимальным приращением 1/65536), изменяя только число частиц в плоскости битов.

[1] **А именно, одну из двух плоскостей и пять из десяти N/VONN соседей.**

8.5. Под вашу ответственность!

Слово соседства, такое как CENTER, является в действительности сокращением для целого ряда *переменных состояния* - по одной на каждую клетку массива и каждый такт времени; аналогично, RAND представляет целый набор *случайных переменных,* по одной на каждую клетку и каждый такт. Однако наиболее практичные реализации RAND многократно использует один и тот же механизм (монета, программа в головном компьютере, клеточно-автоматная система в CAM или внешняя цепь) для порождения значения для всех этих переменных.

Не приведет ли такое *совместное* использование одного механизма к некоторым *корреляциям* между случайными переменными? Это действительно так для любого генератора псевдослучайных чисел.[1] Ну что ж, если мы согласны достаточно заплатить, то нельзя ли получить генератор случайных чисел, имеющий "уровень корреляции", гарантированно меньший заданной границы?

К сожалению, корреляция является слишком сложным понятием для того, чтобы ее можно было удовлетворительно определить одним числовым параметром: корреляции, которые являются пренебрежимыми в одной ситуации, в другой могут изменить саму природу эксперимента. К счастью, мы часто можем дать разумную оценку того, может ли какая-то из корреляционных характеристик конкретного генератора шума неблагоприятно повлиять на эксперимент.

Интуитивно, устройство *R* *случайно* возмущает систему *S,* если его динамика настолько *отлична* от динамики *S,* что у последней нет эффективного способа предсказать, что *R* сделает в следующий момент. Таким образом, для того чтобы устройство *R* можно было использовать в качестве источника шума для S, оно не должно быть хитрее, чем кто-либо еще в мире (это был бы *идеальный* случайный источник); оно должно быть только неожиданным для *S* - при этом система *S* может быть совсем тупой или слишком занятой другими делами, чтобы иметь возможность выделить ресурсы для игры в угадайку. Используя наши знания об ограничениях *S* в этом отношении, мы можем зачастую синтезировать подходящий источник шума *R* поразительно простыми средствами.

[1] Приставка "псевдо" не должна наводить на мысль о каком-то надувательстве. Никто в мире не знает, как сделать "идеальный" генератор случайных чисел, и даже не ясно, является ли эта математическая абстракция корректно определенной.

8.6. Источник шума

Для экспериментов следующей главы мы будем использовать САМ-в как простой генератор шума, порождающий четыре значения 0, 1, 2 и 3 с равными вероятностями. Содержимое плоскостей 2 и 3 будет перемешиваться аналогично тому, как это делалось в разд. 8.2, и начинаться со случайного зародыша. Из САМ-А этот шум будет видим через окно размером 1x1 в САМ-В, предоставляемое назначением минимальной окрестности &/CENTERS (см. разд. 7.3.2).

Для справки, этот генератор шума определяется следующим образом:

```
CAM-B   N/VONN
                                       : NOISE-BOX
       CENTER NORTH WEST SOUTH EAST
       AND XOR XOR XOR  CENTER' XOR
                                >PLN2
  CENTER' NORTH' EAST' SOUTH' WEST'
       AND XOR XOR XOR  CENTER  XOR
                                >PLN3 ;
```

Отметим термы перекрестных связей (т. е. к центру плоскости 2 и к содержимому плоскости 3 применяется операция XOR) и разное размещение соседей относительно операций AND И XOR.[1]

[1] Кнут [31] комментирует такую суеверную практику.

Глава 9

АНТОЛОГИЯ МЕТОДОВ

Если вы впервые начинаете работу с клеточным автоматом, то наверняка испытаете множество правил, используя, возможно, даже случайно сгенерированные таблицы, просто чтобы увидеть, что они делают. Вам захочется освоиться с идеей о том, что машина клеточных автоматов действительно может транслировать написанное на бумаге правило в полный активности мир.

Сейчас мы начнем продвигаться вперед более целенаправленно, ставя перед собой цели - для начала скромные - и пытаясь при этом увидеть, как они могут быть достигнуты (могут и могут ли) имеющимися средствами. Мы стремимся приобрести некоторый опыт - достигнуть такого уровня, когда, столкнувшись с новой проблемой, мы можем попытаться "разделять и властвовать" за счет сведения ее к решенным ранее.

В этой главе мы исследуем некоторые возможности, предлагаемые окрестностями CAM, используя лишь обычных "компасных" соседей (NORTH, SOUTH и т. д.). Использование псевдососедей будет рассмотрено позже.

9.1. Сохранение частиц

Предположим, что нам нужно правило, которое будет генерировать *гистограмму* содержимого плоскости битов; высота каждого столбца гистограммы будет представлять число единиц, которые содержатся в соответствующем столбце плоскости. Идея состоит в том, чтобы нарисовать горизонтальную линию (основание гистограммы) внизу экрана и дать "символам" (т. е. единицам) возможность падать вниз и скапливаться над этой линией стоблец за столбцом.

Символы, которые должны быть подсчитаны, будут находиться в плоскости 0, а линия в основании гистограммы будет проведена в плоскости 1. Правилом для плоскости 1 будет "без изменений": мы хотим, чтобы линия в основании осталась там, где она есть. Правило для плоскости 0 будет уточнением правила SHIFT-SOUTH (ср. с концом разд. 5.1). Давайте начнем с совершенно чистой плоскости 1, за исключением строки единиц внизу - нашей линии в основании гистограммы. Плоскость 0 может быть заполнена случайно или любым паттерном, который вы предпочитаете.

Наиболее важным моментом этого упражнения является то, что мы не хотим потерять или приобрести никаких симво-

лов при их передвижении. Поскольку предполагается, что все сдвигается вниз с постоянной скоростью, то каждая клетка может просто копировать содержимое клетки над ней, будучи уверенной, что ее собственное содержимое будет скопировано клеткой снизу. Однако как только мы введем помехи, эта слепая вера больше не будет действовать; клетка под нами может быть уже заполненной, и *мы* не узнаем, будет ли клетка *под этой* в состоянии принять ее содержимое. Чтобы избежать потери битов информации, клетка будет должна сменить состояние только тогда, когда (а) она знает, что смена состояния возможна и (Ь) она знает, что ее соседи тоже это знают. Следовательно, мы будем должны установить протокол обмена информацией между клетками.

Прежде чем беспокоиться о помехах, т. е. о растущей гистограмме, к которой будут прилипать биты, давайте решим из *локальных* соображений, как двигаться в этой среде, сохраняя частицы. Уместен вопрос: какой участок массива видим одновременно, скажем, мной и моим северным соседом? Этот и *только этот* участок может быть предметом переговоров между нами двумя. Так как мы имеем дело с одномерной системой (каждый столбец массива рассматривается независимо от других), то участок перекрытия состоит в точности из двух клеток, т. е. CENTER и NORTH для меня и CENTER и SOUTH для моего северного соседа. Допустимым правилом, сохраняющим символы, будет следующая альтернатива.

> Если я пуст, а мой *северный* сосед заполнен, то я должен сделать копию его содержимого.

> Наоборот, если я заполнен, а мой *южный* сосед пуст, то я должен стереть свое содержимое. (Я знаю, что он создаст свою копию).

Заметим, что два эти условия не могут быть справедливы одновременно, а значит, могут быть проверены и обработаны независимо.

> В противном случае я ничего не должен делать.

Правило, переведенное в CAM Forth, имеет вид

```
                        : TAKE?  ( -- F|T)
 CENTER 0=  NORTH 0> AND ;
                        : GIVE?  ( -- F|T)
  SOUTH 0= CENTER 0> AND ;
                        : SAFE-PASS
                 CENTER
```

```
        TAKE? IF DROP NORTH THEN
        GIVE? IF DROP SOUTH THEN
                          >PLNO
                 CENTER' >PLN1 ; \ в плоскости 0
                                 \ без изменений
MAKE-TABLE SAFE-PASS
```

Текущее состояние клетки кладется на стек словом CENTER и передается "как есть" >PLN0 до тех пор, пока одно из двух условий не становится истинным; в этом случае оно модифицируется первым. Например, если истинно TAKE?, то выражение DROP NORTH удалит элемент на стеке и заменит его словом NORTH.

Запустим теперь это правило (плоскость 1 все еще не используется). Отдельные частицы будут падать вниз со скоростью света; в заполненных областях различные части будут двигаться с различными скоростями, сжимаясь и вытягиваясь подобно дождевому червю; всякая сплошная область будет размываться у основания и наращиваться сверху новым материалом; "пузырьки" будут двигаться вверх со скоростью света. Однако все вместе будет в конечном счете сдвигаться вниз - частицы при этом будут *сохраняться.* Заметим, что тот же вид правил действовал бы, даже если бы имелись частицы различных "цветов"; сохранялись бы частицы каждого цвета.

Теперь достаточно добавить новое ограничение, а именно подавить действие правила SAFE-PASS через линию основания гистограммы, проведенную в плоскости 1. Общности ради, мы сделаем так, чтобы подавление работало для всякого препятствия, рассматривая его северный край (т. е. любое место в плоскости 1, где 0 находится непосредственно к северу от 1) как непроницаемую границу. Снова, действительно важно лишь, чтобы обе клетки, участвующие в процессе взаимодействия, использовали эквивалентные критерии для распознавания существования ограничения. Для нижней клетки граница указывается условием

```
CENTER' 0> NORTH' 0= AND ,
```

а для верхней

```
SOUTH' 0> CENTER' 0= AND .
```

Если мы погасим этими дополнительными ограничениями "зеленый свет", даваемый соответственно операторами TAKE? и GIVE?, т.е. если мы заменим TAKE? в приведенном выше правиле на

```
                                        : TAKE? (-- F|T)
          CENTER' 0> NORTH' 0= AND NOT
          CENTER  0= NORTH  0> AND AND
```

(и аналогично для GIVE?), то правило будет обладать требуемым поведением. На рис. 9.1 изображены три стадии получения гистограммы.

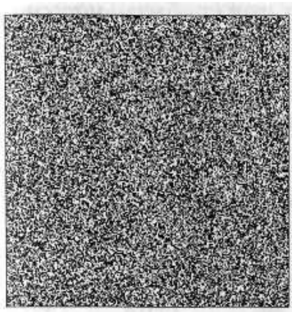
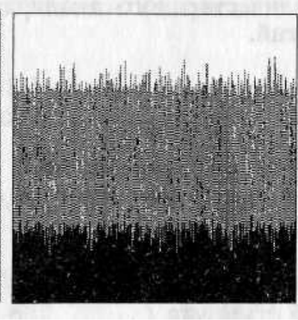




Рис. 9.1. Символы в (а) постепенно конденсируются на линию у основания (Ь), что в конечном счете дает гистограмму (с).

Проблему сохранения частиц и обратимости динамики вообще мы вновь затронем после введения *разделяющихся* окрестностей (таких как окрестность Марголуса), которые предоставляют средства автоматически обеспечить межклеточную координацию для этой цели.

9.2. Дифференциальные эффекты

В эксперименте разд. 6.3 мы сравнили два прогона одного и того же правила, начинающихся с начальных условий, которые различаются только в одном бите. Если возмущение вводится туда, где поведение клеточного автомата активное и сложное, то становится трудно сказать, в какой степени новая история отличается от исходной. Конечно, можно целиком записать две истории и последовательно сравнить их, но это требует большого объема памяти и в конечном счете перемещения большого количества данных для работы по "сличению".

Идеальной ситуацией было бы параллельно запустить две *копии* системы, приготовленные идентично, за исключением небольшого преднамеренного возмущения.[1] В машине клеточных автоматов этот вид эксперимента выполняется очень легко; можно не только использовать две половины CAM (ИЛИ две машины для более сложных предприятий) для двух копий системы, но и сравнить две истории, клетка за клеткой, даже по мере их развития, и изобразить все различия на экране; для более объемного количественного анализа может быть использован счетчик событий.

Мы повторим эксперимент разд. 6.3 с дублями системы, установленными в CAM-A и в CAM-B, как показано ниже:

```
CAM-A N/VONN
                                : TIME-TUNNEL/A
 CENTER NORTH SOUTH WEST EAST
      + + + + { 0 1 1 1 1 0 }
            CENTER' XOR >PLN0
                 CENTER >PLN1  ;
MAKE-TABLE  TIME-TUNNEL/A

CAM-B N/VONN
                                : TIME-TUNNEL/B
 CENTER NORTH SOUTH WEST EAST
      + + + + { 0 1 1 1 1 0 }
            CENTER' XOR >PLN2
                 CENTER >PLN3  ;
MAKE-TABLE  TIME-TUNNEL/B
```

Заметим, что имена соседей одни и те же в двух половинах машины: посылается ли правило в CAM-A ИЛИ В CAM-B, решается словами-"диспетчерами" (>PLN0 и >PLN1 относятся к столбцам таблицы в CAM-A, >PLN2 и >PLN3 в CAM-B). Физические соседи соединяются проводами посредством назначений CAM-A N/VONN и CAM-B N/VONN.

Вторая половина этой программы могла бы быть просто заменена словом B=A, которое дублирует как таблицы, так и отобранные окрестности.

Для того чтобы показать на экране *различия* между соответствующими плоскостями, а не сами плоскости, мы заменим цветовую карту, которую использовали до сих пор (ECHO-MAP, разд. 7.7), надлежащим образом модифицированной картой. В

[1] В физических экспериментах это обычно трудно сделать, так как внешние возмущения по-разному воздействуют на две копии и могут при долгой работе подавлять эффекты, которые требуется заметить.

DIFF-MAP вместо сообщения нам, что бит в плоскости 0 *включен,* зеленый цвет будет указывать, что этот бит *отличается* от бита в плоскости 2 (т. е. гомологичного бита в CAM-в); аналогично голубой цвет будет отмечать различия между плоскостями 1 и 3, а красный будет отмечать те клетки, где различия возникают в *обеих* плоскостях. При таком кодировании цветом на экране ничего не появляется до тех пор, пока две истории идентичны; чтобы увидеть по крайней мере тень того, что происходит, мы передадим содержимое плоскости 0 интенсивностью сигнала.

```
                         : DIFF
          ALPHA  BETA  XOR ;     \ плоскость 0 в сравнении
                                 \   с плоскостью 2
                         : DIFF'
          ALPHA' BETA' XOR ;     \ плоскость 1 в сравнении
                                 \   с плоскостью 3
                          : DIFF-MAP
            ALPHA >INTEN         \ тень пл. 0
DIFF      DIFF'     AND >RED     \ отличается от
                                 \  обеих плоскостей
DIFF      DIFF' NOT AND >GREEN   \ отл. только в пл. 0
DIFF NOT  DIFF'     AND >BLUE ;  \ отл. только в пл. 1

MAKE-CMAP DIFF-MAP
```

Если бы мы хотели *подсчитать* различия, то передали бы значение DIFF слову >INTEN и считывали бы значение счетчика после каждого шага, как объясняется в разд. 7.7.[1]

Теперь мы готовы к работе. Так же как мы делали это на рис. 6.4, мы начнем с большого круга (на этот раз в *обеих* парах плоскостей), выполним 4000 шагов и остановимся. До сих пор обе копии развивались идентично, как можно судить по отсутствию каких-либо цветов на экране (слабое серое изображение на экране, порожденное битом интенсивности, позволяет нам контролировать, что происходит в CAM). Теперь мы заменим один бит в плоскости 0 (пятнышко превратится в ярко-зеленое, как только в одну из двух копий будет внесено возмущение) и двинемся назад во времени. Раковая опухоль распространит свои отростки и начнет обволакивать круг (фото 4); как замечено в разд. 6.3, внутренность круга защищена непроницаемым барьером и на нее возмущение не повлияет.

[1] Так как, согласно этому правилу, плоскость 1 отстает от плоскости 0 на один шаг и аналогично для плоскостей 3 и 2, то нет никакого смысла в вычислении *обоих* различий.

9.3. Соединение двух половин

Применив CAM-A и CAM-B как две независимые системы, мы сейчас готовы и к эксперименту, в котором обе половины CAM соединены в единую динамическую систему.

Мы смоделируем колонию *трубчатых червей;* эти животные имеют щупальца, которые выглядят как нежные цветы, но при малейшем возмущении "цветок" втягивается назад в трубку и ждет около минуты перед тем как появиться вновь. Наши черви будут настолько чувствительны, что для того, чтобы заставить их спрятаться, достаточно возмущения, созданного *n* активными соседями (мы оставляем за собой право поиграть параметром *n).*

"Часы", по которым червь вычисляет время своего пребывания в убежище, будут представлены состоянием половины клетки в CAM-B. В CAM-A содержимое плоскости 0 будет кодировать статус червя (1="активен", о="в укрытии"); плоскость 1 будет чувствовать стимулы и поднимать тревогу, когда имеется в наличии достаточное число активных соседей. Последовательность действий такова:

1. Червь вытянул щупальца; его таймер устанавливается на нуле; детектор стимулов непрерывно отслеживает число активных соседей.

2. Если это число превышает порог, то объявляется тревога в плоскости 1.

3. При обнаружении тревоги таймер устанавливается в положение 3.

4. Червь втягивается в трубку, а таймер переходит в положение 2.

5. Отсчет продолжается: ... 1, о.

6. Червь появляется из укрытия, когда обнаруживает о. Таймер остается в состоянии о, и мы возвращаемся в положение 1.

Конечно, этот рецепт допускает разные интерпретации; приведем нашу:

```
CAM-A N/MOORE &/CENTERS
                                   : 8SUM
        NORTH SOUTH WEST EAST        \ складывать стимулы
```

```
  N.WEST N.EAST S.WEST S.EAST
             + + + + + + +
                                ALARM
  8SUM   { 0 0 1 1 1 1 1 1 1 }      \ звенеть, если два или более червей
                                TUBE-WORMS
  &CENTERS { 1 0 0 0} >PLNO         \ появиться, если
                                    \ время вышло
              ALARM > PLN1          \ посылает сообщение
                                    \   о тревоге туда, где
                                    \   CAM-B может его обнаружить
CAM-B N/MOORE &/CENTERS
                                TIMER
     &CENTER &CENTER' AND IF        \ если "червь высунулся"
                                    \   и "тревога", то
                      3 ELSE        \   установить таймер на 3,
    CENTERS { 0 0 1 2 } THEN        \   иначе сбросить число
                        >PLNB
MAKE-TABLE TUBE-WORMS
MAKE-TABLE TIMER
```

&CENTER означает "бит CENTER другой половины CAM;" для CAM-A это означает бит плоскости 2, но для CAM-B это означает бит плоскости 0.

Стимулы в CAM-A не могут быть обнаружены таймером в CAM-B, так как назначение минимальной окрестности &/CENTERS разрешает последнему доступ только к *центральным* битам CAM-A. Мы должны пройти через двухстадийный процесс: (a) в CAM-A, где мы имеем доступ ко всем девяти соседям, мы подсчитываем стимулы, сравниваем их число с порогом и запоминаем результат сравнения в центральной клетке плоскости 1;(Ь) на следующем шаге CAM-B берет результат оттуда.

Начнем этот эксперимент со случайной конфигурации во всех четырех плоскостях, так что относительная фазировка циклов червей случайна. По прошествию короткого времени станут видны когерентные фазовые волны, которые будут расти и сливаться, создавая узор, напоминающий передвигающиеся песчаные дюны (рис. 9.2a).

Правила этого вида очень чувствительны к деталям цикла обратной связи. При п=2, как и выше, узор развивается очень быстро и волны весьма малы. При *n=Ъ* волны уширяются и их движение замедляется (рис. 9.2Ь); вскоре развиваются самоподдерживающиеся центры активности, принимающие обычно форму спаренных спиралей ("бараньи рога"). При п=4 большие области в конце концов застывают в фиксированной фазе, и интересной активности почти не остается. Наиболее интересное поведение получается, возможно, при следующем условии "тревоги":

8SUM { 0 0 1 0 1 1 1 1 1 } ,

которое является разновидностью "стабилизирующей" версии с м=3 (см. ANNEAL в разд. 5.4); это показано на рис. 9.2с. Те же три правила проиллюстрированы на фото 5-7.



Рис. 9.2 Пространственные реакции: (а) и (Ь) используют различные значения параметра обратной связи, тогда как (с) является вариантом (Ь) с немонотонным порогом.

Системы этого рода дают хорошие модели для конкурентных/кооперативных явлений некоторого типа, таких как знаменитая реакция Белоусова - Жаботинского [75].

Существует четкая аналогия между этой моделью и моделью "возбуждения нейронов", описанной в разд. 6.1. В обоих случаях стимул обеспечивается наличием определенного числа соседей, а отдельные клетки характеризуются некоторым временем восстановления. Однако есть существенная разница в природе цикла обратной связи. В модели нейрона обратная связь положительна: только возбуждение нейронов может приводить к новым возбуждениям. Подобная система имеет, вообще говоря, два устойчивых режима: (а) "активность, которая порождает активность" и (Ь) "бездействие, которое порождает бездействие".

В модели колонии червей обратная связь *отрицательна* на коротком отрезке: наличие червей скорее *подавляет,* чем *стимулирует* появление большего их числа в непосредственной окрестности. Однако линия задержки, представленная таймером, создает такую ситуацию, что для определенных частот и длин волн обратная связь положительна ("Если черви сейчас здесь *присутствуют,* то некоторое время их будет мало в ближайшей окрестности, а это означает, что в кольце, непосредственно примыкающем к этой окрестности, возмож-

ность их появления *возрастет".)* Характерные периоды и длины волн, которые наблюдаются в этой модели, соответствуют пространственно-временным паттернам, для которых коэффициент обратной связи приблизительно равен единице.

9.4. Генетический дрейф

Используя генератор случайных чисел, мы можем предпринять первую попытку моделирования типа поведения, который очень интересен сам по себе и к тому же является важной компонентой более сложных рецептов. Мы имеем в виду *диффузию.* Мы хотим представить единицу как "частицу" в "пустом пространстве" нулей; мы хотим, чтобы на каждом шаге эта частица делала случайный шаг в одном из четырех возможных направлений.

Идея очень проста. Мы можем поместить частицы в плоскости 0, а источник шума в плоскостях 2 и 3 (где источник шума описан в разд. 8.6). Частица будет наблюдать за двумя случайными битами, на которых она расположена, и использовать их как подбрасывания монеты, чтобы решить, двинуться ей вверх или вниз, вправо или влево. Проще всего записать следующее интуитивно очевидное правило:

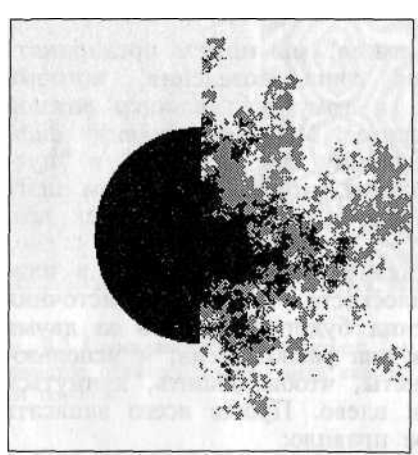
```
                                      : NAIVE-DIFFUSION
&CENTERS { NORTH SOUTH WEST EAST }
                                >PLNO ;
```

Случайное значение &CENTERS (это двухблочный вариант &CENTER и &CENTER) изменяется в интервале от 0 до 3 и мы используем его для выбора одного из четырех направлений.

Давайте испытаем это правило, начиная со сплошного диска. Черная и белая области диффундируют друг в друга: диск разбивается на языки огня (на разделенном экране на рис. 9.3а показаны ситуации "до" и "после"). Однако даже после долгого ожидания мы получим на экране хлопья, а не однородную смесь. Если мы сравним этот паттерн с тем, который изображен на рис. 9.3b (который был порожден более сложным алгоритмом диффузии, обсуждаемым в разд. 15.1), то увидим, что что-то не так. Более того, если мы шаг за шагом *подсчитаем* частицы, то увидим, что их число флуктуирует: частицы не сохраняются.

Все это не должно быть неожиданным. В клеточном автомате область влияния клетки не распространяется дальше, скажем, ее южного соседа: она не может напрямую "отложить" там частицу, как объясняется в разд. 5.1; все, что клетка может сделать, чтобы двинуть что-то на юг, так это сбросить

свое собственное содержимое и "скопировать" содержимое ее северного соседа. Без установления протокола взаимодействия между клетками (см. разд. 9.1) потери и дублирования неизбежны. Все, на что мы можем надеяться, так это то, что частицы будут сохраняться *в среднем.*



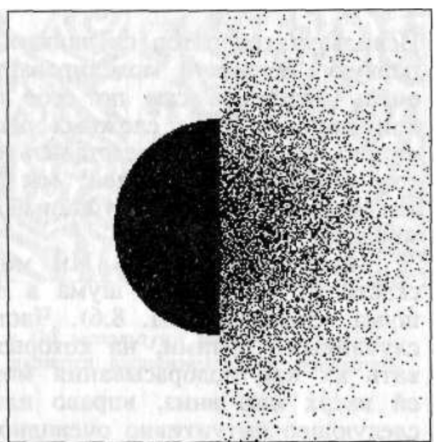


Рис. 9.3. (а) Псевдодиффузия, полученная согласно правилу "копируй со случайного соседа", и истинная диффузия (Ь). Оба рисунка изображены на разделенном экране, показывающем половину "до" и половину "после"; исходная конфигурация - диск.

Хотя мы и не достигли первоначальной цели, но правило, которое мы получили, действительно дает разумную модель *генетического дрейфа* [29, 12]. Гены действительно диффундируют посредством создания своих копий, а в ситуации относительного равновесия между смертностью и рождаемостью каждый наличный ген оставляет в среднем одну свою копию в каждом поколении. Описанный выше эксперимент можно быть повторить, используя четыре, а не две "разновидности" генов, используя оба бита плоскости САМ-А ДЛЯ кодирования того, какая из четырех разновидностей представлена данной клеткой (САМ-в по-прежнему используется как генератор шума). Правило, конечно, имеет вид

```
                                   GENETIC-DRIFT
«.CENTERS { NORTHS SOUTHS WESTS EASTS }
                              >PLNA
```

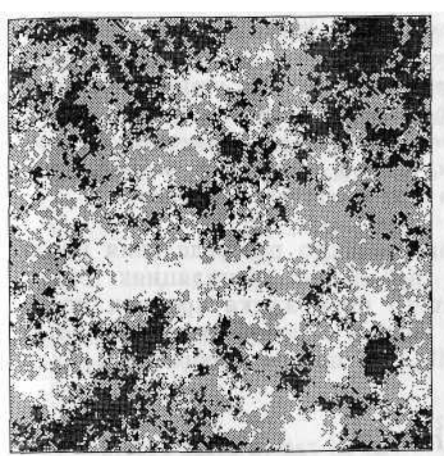
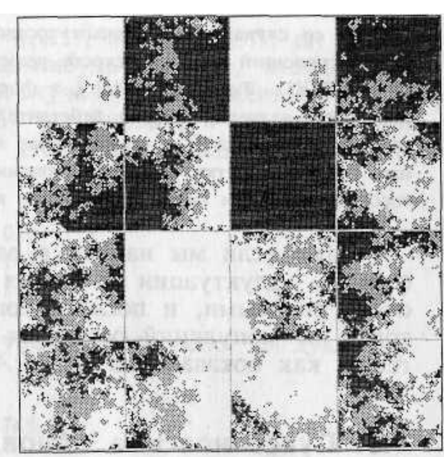



Рис. 9.4. Генетический дрейф: (а) большая популяция проявляет незначительные флуктуации ее генофонда; (Ь) в изолированных популяциях может возникать постоянное обеднение генофонда.

Если мы начнем с равномерного (случайного) пространственного распределения четырех видов генов, то достигаемое некоторое время спустя стационарное состояние представляет собой пятнистое распределение (рис. 9.4а и фото 8): одни и те же виды могут быть в избытке или в недостатке в различных областях. Наконец, изучим генетический дрейф в малых изолированных популяциях. С этой целью мы разделим экран на большое количество квадратов; это достигается вычерчиванием решетки в плоскости 1 и созданием правила для плоскости О, интерпретирующего эту решетку как барьер для передачи генов (так как для генов остается доступной только плоскость О, у нас будут только два вида генов). Приведенное выше правило NAIVE-DIFFUSION модифицируется, чтобы учесть решетку, следующим образом:

```
                                        6ENE-CIDE
                           &CENTERS
 { NORTHS SOUTHS WESTS EASTS }
 DUP 1 > IF DROP CENTER THEN            \ используйте CENTER, если
                                        \   сосед - это решетка
                              >PLNO
                      CENTER' >PLN1  ; \ оставить решетку на месте
```

Наряду со случайно выбранным соседом, скажем NORTH, мы также выбираем соответствующий бит в плоскости решетки, т. е. NORTH' (два бита объединяются в NORTHS). Выражение DUP 1 > формирует копию того, что мы выбрали, и проверяет, является ли оно действительно частью решетки (состояния 2 и 3 означают "решетка"); в таком случае мы сбрасываем его и заменяем на текущее значение клетки. Другими словами, решетка действует как зеркало, в котором каждый ген видит свою копию в этом конкретном направлении.

Даже если мы начнем с равномерного распределения генов О и 1, флуктуации в таких небольших популяциях будут очень сильными, и после какого-то промежутка времени некоторые из популяций останутся только с одной разновидностью генов, как показано на рис. 9.4Ь; эта потеря необратима.

9.5. Пуассоновское обновление

Обычный клеточный автомат является по определению системой с дискретным временем: клетки меняют свое состояние в целочисленные моменты времени. Часто бывает полезно рассмотреть модели, в которых изменение состояния клетки (которая по-прежнему определяется дискретной функцией над областью состояний) может происходить в произвольный момент на всей *непрерывной оси времени*.

Важнейшим примером служат пуассоновские процессы, в которых события происходят случайно при равномерном распределении во времени, и, следовательно, вероятность, что следующее событие появится между моментами времени t и $t+dt$, равна $Xe{\sim}^{xt}dt$. В *пуассоновском клеточном автомате* обновление состояния каждой клетки управляется независимым пуассоновским процессом. Вероятность того что при таком обновлении две клетки будут обновлены *точно* в один и тот же момент, равна нулю; поэтому при написании правила обновления для клетки несомненно можно предположить, что эта клетка является *единственной* клеткой, которая может изменить состояние; с учетом этого предположения протоколы взаимосвязей, которые могут быть необходимы, чтобы гарантировать выполнение некоторых ограничений (см. разд. 9.1), можно значительно упростить.

В этом разделе мы сначала дадим простой пример системы, для которой эти два метода обновления дают радикально разное поведение; затем мы покажем, как пуассоновское обновление можно эмулировать с какой угодно точностью обычным клеточным автоматом.

Рассмотрим следующую проблему "эрозии почвы". Кусочек почвы, представленный единицей, останется на месте, если с северной стороны от него есть хоть чуть-чуть почвы (т. е. в

одном из трех северных соседей NWEST, NORTH, N.EAST); он также сохранится, если почва есть где-нибудь на юге, западе или востоке от него. В первых трех случаях, изображенных ниже, почва в центре окрестности 3x3 "устойчива", в то время как в четвертом она "незакреплена" (так как никакая из трех его восточных позиций не занята):

```
101   010   010   010
010   ПО    111   НО
010   001   010   НО
```

Незакрепленная почва будет унесена согласно следующему правилу:

```
                               : STABLE
N.WEST NORTH N.EAST OR OR
S.WEST SOUTH S.EAST OR OR
N.WEST WEST  S.WEST OR OR
N.EAST EAST  S.EAST OR OR
            AND AND AND ;
                               : SOIL
   CENTER STABLE AND >PLNO ;
```

Слово STABLE является бит-маской; от него и текущего состояния клетки берется операция AND. ЕСЛИ значение этой маски единица, то почва остается на месте; если о, то она выветривается.

Если вы запустите это правило, начиная со сплошной почвы, то ничего не произойдет. Если вы удалите в некоторых местах изолированный кусочек почвы, то в дальнейшем эрозия не возникнет. Если вы продолжите удаление кусочков случайно, то в конечном счете получите места, где удалены два или три примыкающих кусочка; в зависимости от формы такой "ямки" стенки могут "рушиться", увеличивая саму ямку. Пока количество удаленной почвы остается ниже определенного критического уровня (около 17%), такие обрушения обычно самозалечиваются и почва в целом остается устойчивой (рис. 9.5а).

Однако когда доля удаленной почвы превышает этот уровень, некоторые из ямок проявляют неограниченный рост - они становятся *центрами образования зародышей* [66], и в конечном счете вся почва выветривается (рис. 9.5b).[1]

[1] Переход очень резкий, т. е. вероятность цепной реакции сохраняется близкой к нулю, даже когда доля удаленной почвы становится очень близкой к порогу, а затем быстро сдвигается к значениям, близким к единице, как только порог превышается.

Прекратите случайное удаление почвы как раз перед достижением критического порога, когда она все еще устойчива, и наблюдайте ситуацию. Если теперь вместо случайных действий вы позаботитесь о том, чтобы оставить кусочки, которые существенны для устойчивости их соседей, то сможете удалить до *половины* почвы без запуска цепной реакции (устойчив, например, шахматный узор). Надежным эмпирическим правилом для одинокого "освоителя целины", только что сброшенного на парашюте на участок, было бы удалять участок почвы (распахивать его), только если почва есть *непосредственно* на севере, юге, западе и востоке от него, т. е. в клетках NORTH, SOUTH, WEST и EAST; вы можете проверить, что это не создаст каких-либо неустоичивостеи. Правило может быть закодировано следующим образом:

```
                              : SAFE (-- 0|1)
   NORTH  SOUTH  WEST  EAST
                 AND AND AND ;
                              : NAIVE-DEVELOP
     CENTER STABLE AND
        SAFE NOT AND  >PLNO ;
```

При этом SAFE NOT снова используется как маска. Как и прежде, кусочек почвы сохранится, если он "устойчив", и если не сочтено "безопасным" удалить его.

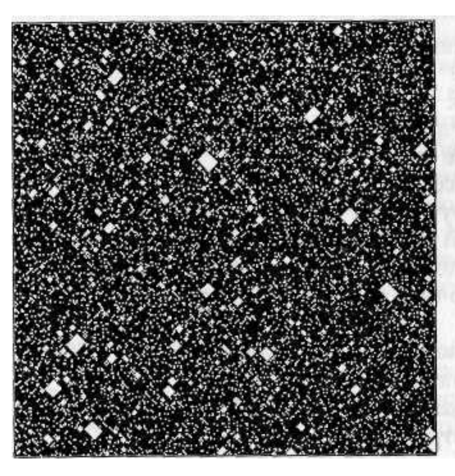
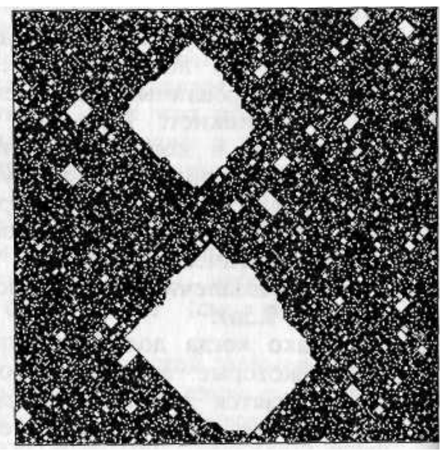



Рис. 9.5. Образование зародышей: (а) до тех пор, пока удаление почвы не превышает критического уровня, конфигурация, как правило, остается устойчивой; (Ь) выше этого уровня центры образования зародышей возникают и растут неограниченно.

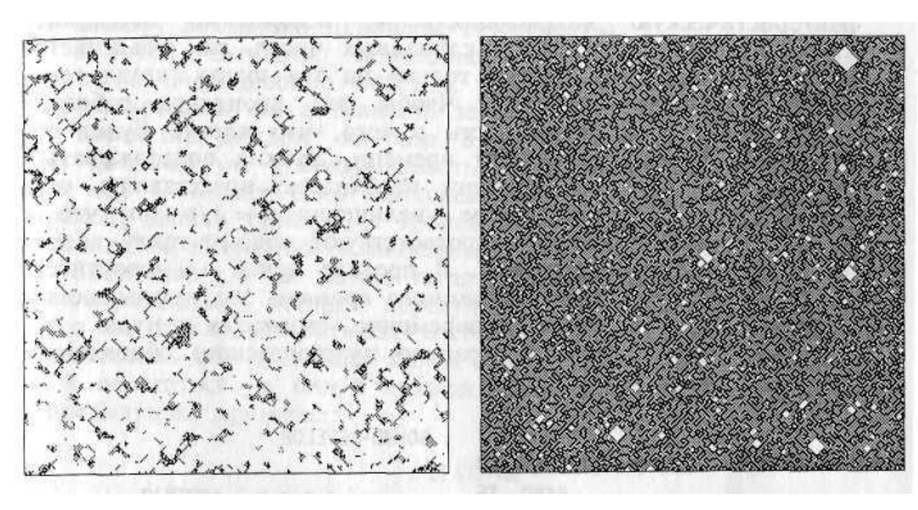


Рис. 9.6. Освоение целины: (а) при обычном обновлении инициализация разрушительных цепных реакций весьма вероятна; большие белые области являются растущими очагами эррозии; (Ь) законченное освоение, достигнутое при пуассоновском обновлении.

Однако если вы запустите это правило на обычном клеточном автомате, начиная с устойчивой конфигурации рис. 9.5а, то можете получить неожиданный результат. Правило применяется ко всем клеткам одновременно. Если два не подозревающие о намерениях друг друга освоителя сосредоточатся одновременно на прилегающих участках земли, то они могут "подсечь" друг друга и в конечном счете обнаружат себя окруженными пустыней (рис. 9.6а): игра NAME-DEVELOP безопасна, только когда в каждый момент она играется одним человеком. Чтобы исключить неприятности, когда на участке работает больше чем один освоитель, должны быть введены более сложные формы подавления эрозии.

Для пуассоновского обновления, напротив, возможность неприятности, подобной предыдущей, нулевая, и результатом долговременного развития является прерия, покрытая беспорядочной мозаикой коттеджей (рис. 9.6Ь).

Это был, конечно, экстремальный пример. Тем не менее обновление в случайные моменты времени, как правило, полезно для систем, в которых модели с синхронным обновлением привнесли бы ложные, нежелаемые симметрии.

Чтобы эмулировать пуассоновский процесс, имеющий характеристическую интенсивность A, необходимо обеспечить каждую клетку генератором случайных чисел, как объясняется в гл. 8, и заменять клетку только на тех шагах, когда генератор возвращает единицу. Чтобы эта процедура работала удовлетворительно, вероятность p того, что клетка будет обновлена в некоторый момент времени, должна поддерживаться низкой, для чего ось времени необходимо представлять себе растянутой с коэффициентом увеличения κ = Л/р (так что κ шагов клеточного автомата соответствуют одному такту времени в моделируемой системе). В пределе при p —* 0, вероятность того, что за один такт *системного времени* две примыкающие клетки будут обновлены одновременно, стремится к нулю.

Для данного примера правило пуассоновского обновления, конечно, имеет вид

```
                    : SOUND-DEVELOP
CENTER  STABLE  AND
            RAND  IF
SAFE  NOT AND  THEN
           >PLNO ;
```

где RAND является выходом генератора случайных чисел, который можно было бы разместить в САМ-в. Источник шума из разд. 8.6 не пригоден для этой цели, так как каждая из двух плоскостей дает вероятность 1/2 и даже взятие операции AND от их содержимого дало бы вероятность 1/4. Для метода из разд. 8.2, используя при этом девять, а не пять соседей в окрестности, можно получить весьма малые вероятности, до 1/512, что достаточно во многих случаях. В разд. 15.6 мы обсудим метод получения более широкого динамического диапазона генератора случайных чисел.

Другой способ исключить "недоразумения" между освоителями целины состоит в том, чтобы разрешить обновление лишь каждой второй клетки - в форме шахматной доски - за один шаг, а остальных клеток - на следующем шаге. Этот метод обсуждается в разд. 11.6 и используется в гл. 17.

9.6. Асинхронные детерминированные вычисления

В экспериментах по генетическому дрейфу разд. 9.6 каждая клетка была безусловно вынуждена делать копию одного из своих соседей, выбранного случайно. Можно представить себе и большее число возможных исходов таких конфронтации; например, можно позволить клетке сохранять свою индивидуальность, пока она не встретится с оппонентом, который в неко-

тором смысле "сильнее". Правила этого вида были предложены нашему вниманию Дейвидом Гриффитсом.

В этой ситуации монотонное ранжирование состояний клетки с очевидностью ведет к насыщению; по прошествии долгого времени вся активность затухает (подобно эксперименту "свечи под дождем" из разд. 8.1). Давайте взамен рассмотрим *циклическую* упорядоченность последовательности состояний, где 0 может быть вытеснен только единицей, единица двойкой, двойка тройкой, а тройка нулем - ни одно состояние не является абсолютно сильнейшим.[1]

Состояние клетки можно тогда интерпретировать как *фазовую переменную,* которая нерегулярно, но неуклонно проводится по циклу из четырех состояний. Наше правило будет использовать обе плоскости CAM-A для кодирования фазы клетки и полагаться на CAM-B в качестве источника шума, как и в предыдущих примерах:

```
: BEATS-ME
      CENTERS { 0 1 2 3 } = ;
: CYCLIC-RANK
      &CENTERS
      { NORTHS SOUTHS WESTS EASTS }
      DUP BEATS-ME NOT IF
      DROP CENTERS THEN
      >PLNA ;
```

Слово BEATS-ME сравнивает ранг выбранного соседа с рангом самой клетки.

Аналогичные механизмы могут быть использованы для обеспечения *причинной согласованности* в параллельных компьютерах с асинхронным обновлением.

Рассмотрим, например, клеточный автомат, клетки которого обновляются несколькими независимо думающими "рабочими". Рабочих может быть меньше, чем клеток, и они могут делать паузы в случайные моменты времени, они могут утомиться от работы всегда в одном месте или над одной задачей, их работа может быть приостановлена или прервана внешними факторами и т. д. Не стоит надеяться, что все клетки будут обновлены одновременно, скажем по свистку бригадира.

Несмотря на все это, мы хотим, чтобы асинхронные системы развивались *изоморфно* системе, которая заменяется строго синхронным образом. Если клетка находится временно впереди других, то она должна иметь состояние, которое имела бы, ес-

[1] Это похоже на хорошо известную игру двух лиц "камень, ножницы, бумага", где бумага оборачивает камень, камень притупляет ножницы, а ножницы режут бумагу.

7-1608

ли бы все другие клетки обновлялись "в ногу" с ней. Хотя расписание событий во времени не будет определено заранее, мы хотим, чтобы в каждом месте нужные события происходили в нужной последовательности; отсюда следует, что правильная причинная связь между событиями, возникающими в различных местах, должна быть сохранена.

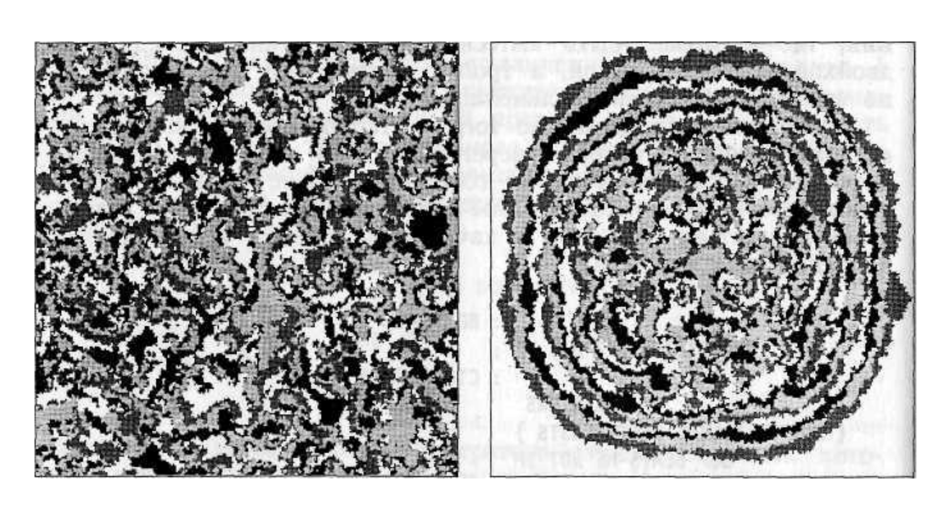


Рис. 9.7. Фазовые волны в циклически упорядоченных системах: (а) от равномерной случайности и (Ь) от небольшого пятна случайности.

Это не только может быть достигнуто, но и достигнуто относительно простыми средствами. Решение (которое приложимо к компьютерам, имеющим произвольную структуру - не только к клеточным автоматам) опирается на следующие три рецепта[1]:

a. Состояние клетки *буферизовано,* т. е. новое состояние клетки вычисляется и запоминается в отдельном регистре, в то время как текущее состояние остается видимым для соседей. Только когда все соседи вычислили их собственные новые состояния, текущее состояние будет заменено новым. Заметим, что для этого метода единственный шаг обновления исходной систе-

[1] Это каноническое решение проблемы синхронизации обсуждается в [56].

мы заменится в асинхронной системе *циклом обновления,* состоящим из нескольких шагов.

b. В дополнение к регистру *текущего состояния,* который общедоступен, и регистру *следующего состояния,* который является личным делом клетки, клетка обеспечивается *фазовой переменной* (в данном случае целым числом от 0 до 3), которая указывает на ее продвижение в цикле обновления. Эта переменная должна быть также видимой соседям, чтобы сделать возможной межклеточную координацию.

c. Клетка никогда не продвинется вперед на один шаг, если она уже находится на один шаг впереди по крайней мере от одного из ее соседей. Это предписание обосновывается ниже.

Здесь мы будем непосредственно касаться лишь пунктов Ъ и с, т. е. мы реализуем только фазовую переменную и правило ее эволюции (в разд. 12.8.3 мы приведем пример, в котором для того, чтобы следовать такому асинхронному порядку, выполняется определенное вычисление). Если мы начнем с плоской "фазовой поверхности", распространенной на весь массив, т. е. если все клетки первоначально имеют одинаковую фазу, то позволим этой поверхности деформироваться с появлением гор и долин по мере продвижения во времени в асинхронном режиме, но никогда не допустим ее "разрывов", соответствующих потере причинной согласованности.

С точки зрения каждой клетки четыре значения фазовой переменной соответствуют следующим стадиям диалога с ее соседями:

0: Я по-прежнему наблюдаю за некоторыми из вас (не изменяйте пока вашего состояния); некоторые из вас могут наблюдать за мной (я не могу пока изменить мое состояние).

1: Я больше не наблюдаю за какой-либо из вас (продвигайтесь вперед и, если хотите, объявляйте новое состояние - я уже использовала свое старое состояние для вычисления моего следующего, которое запомнила в секретном месте); некоторые из вас могут по-прежнему наблюдать за мной (я буду хранить свое новое состояние для себя и по-прежнему изображать старое).

2: Я не наблюдаю за какой-либо из вас. Никакая из вас не должна наблюдать за мной (я могу теперь объявить мое новое вычисленное состояние в общедоступном регистре, заменив предыдущее).

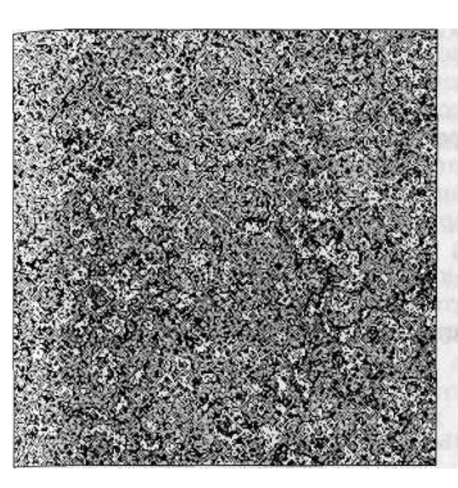
3: Я вновь наблюдаю за вами (если вы объявили уже свое новое состояние, храните его там; если нет, то ждите, пока вам не надоест - я буду ждать столько, сколько необходимо); никакая из вас не должна теперь наблюдать за мной.

По подсказке генератора асинхронных импульсов клетка проверит фазы ее соседей и выполнит переход к следующему этапу указанного выше правила, только если это допускает ситуация; в противном случае она запомнит время. В отличие от примера разд. 9.5, мы не касаемся здесь вопроса, могут ли примыкающие клетки быть приведены в активное состояние одновременно. Там мы хотели моделировать асинхронную систему синхронной; здесь мы желаем делать в точности обратное, а возвращение время от времени к синхронизму не повредит.

Мы будем использовать CAM-A для фазовой переменной, а источник шума в CAM-в для пуассоновских часов[1]. В CAM Forth правило выглядит следующим образом:

```
                       : STIMULUS? ( - F|T)
         &CENTERS 0- ;    \ Время действовать!
                       : CENTERS+1 ( - след. этап)
   CENTERS { 1 2 3 0 } ;
                       : TRANSIT? ( - F|T)
  NORTHS CENTERS   =      \ Я на одном уровне с
                          \  северным соседом
   NORTH CENTERS+1 = OR   \  или отстала на шаг?
  SOUTHS CENTERS   =      \ То же для юга
  SOUTHS CENTERS+1 = OR
   WESTS CENTERS   =      \ И т.д.
   WESTS CENTERS+1 = OR
   EASTS CENTERS   =
   EASTS CENTERS+1 = OR
         AND AND AND ;
                       : ASYNC
STIMULUS? TRANSIT? AND IF  \ Если я могу (и должна),
        CENTERS+1 ELSE     \  то продвинусь вперед,
          CENTERS THEN     \  иначе, пока!
                >PLNA ;
```

[1] Генератор импульсов, распределенных по Пуассону. - *Прим. перев.*



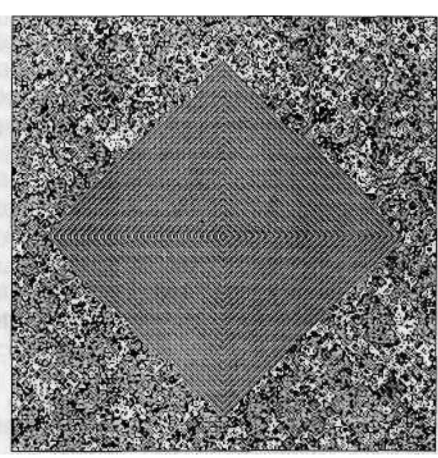


Рис. 9.8. (а) Типичные линии уровня фазовой "поверхности". В (Ь) поверхность была "пригвождена" в одном месте постоянной остановкой часов одной клетки.

Слово CENTERS+1 просто определяет следующий этап в цикле. Слово TRANSIT? проверяет, есть ли какая-нибудь помеха продвижению фазы вперед. Для используемого нами постоянно источника шума "рабочего" на каждом шаге можно побудить с вероятностью /»=1/4 (словом STIMULUS), для того чтобы проследить за порядком; различные уровни вероятности можно получить регулировкой генератора случайных чисел.

Мы начнем с конфигурации, все клетки которой имеют фазу о, соответствующей ситуации, где все клетки демонстрируют их текущее состояние и ни одна из них не вычислила свое следующее (напомним, что компонента *состояние* системы не представлена явным образом в настоящем примере; представлена только *фазовая* компонента). По мере работы генератора случайных импульсов некоторые клетки перейдут к фазе 1. В конечном счете некоторые клетки будут целиком окружены единицами и смогут перейти к фазе 2 (в компоненте состояний системы в этот момент часть массива уже будет демонстрировать новое состояние, впереди остальной части массива) и т. д.

Различные части фазовой "поверхности" будут находиться на различных временных уровнях. Фазовый сдвиг между отдаленными клетками может достигать нескольких полных циклов (один фазовый цикл соответствует одному шагу синхронного автомата, который моделируется асинхронным образом). Мак-

симально возможное разделение по времени составляет один цикл на четыре единицы расстояния: свыше этого значения "тяга" отстающих фаз в окрестности становится непреодолимой, и никакой дальнейший прогресс не возможен до тех пор, пока соседи клетки не подтянутся, уменьшив это отставание.

На рис. 9.8 показано типичное распределение фазы во времени, когда память о первоначальном "плоском" назначении фазы полностью утрачена; на рис. 9.8Ь генератор асинхронных импульсов остановлен для одной клетки[1], и в конечном результате вся система затормаживается и останавливается, раньше для ближайших клеток и позже для более удаленных.

9.7. Одномерные клеточные автоматы

При моделировании одномерной системы ресурсы двумерной машины клеточных автоматов размером *mxn* могут быть перераспределены несколькими способами.

Например:

1. Можно запустить множественные копии системы (по одной в каждой строке массива), начиная каждую с разных начальных условий, как на рис. 10.2. Это может быть полезно для статистических исследований.

2. Систему можно моделировать и в единственной строке массива. В этом случае оставшиеся /и-1 строк можно использовать для записи истории системы в обратном направлении, от самого последнего состояния через все промежуточные до (m-l)-ro шага в прошлом, как на рис. 10.1. Таким способом изображение пространственно-временной истории системы "прокручивается" в режиме свитка по массиву.

3. Можно соединить концы строк в виде спирали, синтезируя таким образом одномерную машину размером *m.n*.

Первый подход реализуется тривиально. Если правило использует только западного, восточного и центрального соседей, то автоматически получается одномерная система. Для окрестностей больших размеров смотрите разд. 9.8.

[1] В действительности это было сделано установкой ее фазы на два шага назад относительно ее соседей.

При втором подходе необходимо пометить какую-нибудь строку (скажем, самую нижнюю) как строку, в которой происходит эволюция; эта пометка,, которую мы будем называть строкой NOW, может быть запомнена в другой плоскости, или может быть получена от аппаратных средств, таких как псевдососед, как мы разъясним ниже. Правило будет считывать эту строку и регулярно воздействовать на клетки, из которых она состоит; в других строках массива правило будет просто "сдвигать на север". Таким образом, самое последнее состояние будет внизу массива, а наиболее раннее, которое все еще видимо, будет сверху; ось времени будет направлена вниз (физики могут предпочесть заменить правило так, чтобы оно сдвигало в противоположную сторону).

В качестве очень простого примера давайте создадим одномерный генератор случайных чисел, используя подход, обсужденный в разд. 8.2. Строка NOW будет помещена внизу плоскости 1, а генератор будет расположен в плоскости 0. Правило имеет вид

```
                        : NOW ( -- 0|1)
                CENTER' ;
                        : ONED-RAND
                 NOW IF     \ если на строке NOW
WEST CENTER EAST OR XOR ELSE \   запустить правило
              SOUTH THEN    \   иначе сдвинуться на
                  >PLNO     \   север
          CENTER' >PLN1 ;   \ Строка NOW остается
                            \  на месте
```

Эволюция этого генератора случайных чисел, который был изучен Стивеном Волфрамом [73], показана на рис. 9.9а.

Если нам не захочется тратить целую плоскость на строку NOW, то можем использовать сигнал, именуемый -VFF, который подается внутренней схемой и имеется на пользовательском коммутаторе.[1] Чтобы создать пользовательскую окрестность, в которой имеется этот сигнал, мы должны использовать, например, назначение дополнительной окрестности &USER, которое питает адресные линии 10 и 11 справочной таблицы с пользовательского коммутатора. На этом коммутаторе мы должны подсоединить перемычку от вывода -VFF К выводу USER 10 CAM-A. Наконец, мы должны назначить имя соседа -NOW адресной линии 10 и определить NOW как дополнение -NOW.

[1] -VFF возвращает 0 для клеток в *нижней строке* (FF равно 255 в шестнадцатиричной системе счисления) и единицу в других местах. Имеются также -VOO (верхняя строка), -HOO (крайний столбец слева) и -HFF (крайний столбец справа).

```
10 == -NOW
                         : NOW ( — 0|1)
               -NOW 1 XOR
```

Перемычка может остаться на месте, даже когда в ней нет необходимости, так как она будет использоваться только во время действия назначения &USER

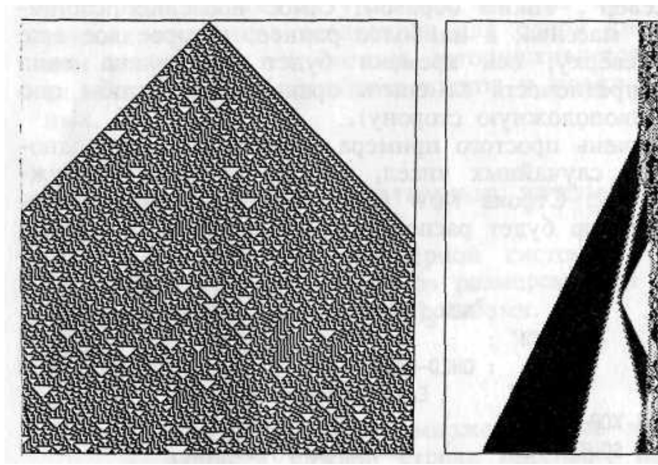
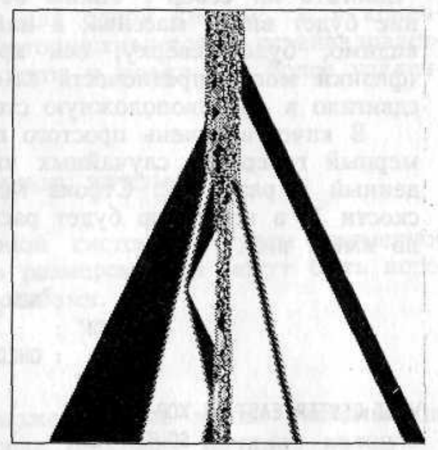



Рис. 9.9. (а) Одномерный генератор случайных чисел. (Ь) Образец пространственно-временной истории от правила SCARVES - одномерного механического микрокосма.

В третьем подходе интуитивно ясно, что мы должны "разрезать" тор (см. разд. 2.2.4), скажем, по левому краю массива и соединить два края разреза после смещения их на одну строку по отношению друг к другу, достигая, таким образом, спирального обвивания. Механизм для достижения этой цели состоит в том, чтобы заставить клетки на правом краю разреза рассматривать как EAST то, что в действительности является N.EAST, а клетки на левом краю рассматривать как WEST то, что в действительности является sWEST. Положение разреза может быть помечено вертикальной линией, запомненной в плоскости или взятой от псевдососедей -ноо и -HFF, имеющихся на пользовательском коммутаторе (см. примечание на стр. 103).

Простое, но очень интересное правило, которое является одномерным аналогом правила SPINS-ONLY разд. 17.3, изучалось Чарлзом Беннеттом. SCARVES является обратимым правилом второго порядка (см. разд. 6.2), которое наблюдает не только за

непосредственными соседями WEST и EAST, но и за вторичными соседями WEST-OF-WEST и EAST-OF-EAST (способы извлекать содержимое этих дополнительных соседей описаны в следующем разделе). Содержимое клетки будет сбрасываться по мере продвижения из прошлого в будущее, если среди четырех ее соседей в данный момент имеется два нуля и две единицы. При работе в одном измерении согласно методу 2 нет необходимости запоминать прошлое состояние клетки (нужное для правила второго порядка) в дополнительной плоскости, так как это состояние уже имеется в примыкающей строке той же плоскости, где оно и может быть обнаружено как сосед NORTH.

Мир клеточного автомата, определенный этим правилом, является настоящим микрокосмом механических явлений обычной и статистической механики (рис. 9.9Ь). Он поддерживает большой набор элементарных "частиц" (аналогичных глайдерам разд. 3.4); некоторые из них маленькие и быстрые, а некоторые большие и медлительные. Частицы могут расщепляться в результате столкновения и возникать как новые комбинации частиц; тем не менее *энергия по Изингу** при столкновениях сохраняется [45]. Если вы начнете с облака "горячего газа" - случайного пятна на чистом фоне - облако будет медленно терять тепло от расширения и испарения, и возникнут различные уровни упорядоченных структур.

9.8. Приемы расширения окрестности

Существует несколько методов для считывания значений вторичных соседей, необходимых для этого правила, и в общем случае для введения в поле зрения клеток, не попадающих внутрь окна 3x3, предоставляемого CAM; мы коснемся всего лишь двух из них, а именно *собирания* и *плиссирования*.

Предположим, что вы хотите "собрать" ваших вторичных соседей, которых мы назовем WEST-WEST и EAST-EAST, в состав окрестности. Вы можете скопировать текущее содержимое плоскости 0, скажем в плоскость 1, а затем сдвинуть плоскость 1 на одну позицию вправо и WEST-WEST будет теперь видима как WEST. Дополнительный шаг, требуемый этим сдвигом, замедляет моделирование, но вы приобрели еще одного соседа. Как всегда, для того чтобы извлечь максимальную выгоду из ресурсов, могут понадобиться заказная окрестность и некоторая изобретательность.

Более эффективный способ получить дополнительных соседей состоит в том, чтобы "плиссировать" массив так, что он

[1] Величина, которая учитывает, сколько нынешних соседей "противоречат" прошлому значению рассматриваемой клетки; см. разд. 6.2, 17.2.

будет более узким в направлении запад-восток и соответственно станет толще; дополнительная толщина обеспечивается второй плоскостью битов. Это иллюстрируется следующей диаграммой, в которой буквы представляют клетки; одна пятиклеточная окрестность задается заглавными буквами. Мы начнем с конфигурации, в которой клетки располагаются в определенном порядке в одной плоскости битов

$$a \quad b \quad c \quad D \quad E \quad F \quad G \quad H \quad i \quad j \quad k \quad l$$

и перепишем эту конфигурацию как такую, в которой те же клетки располагаются в двух плоскостях битов

$$\begin{matrix} a & c & E & G & i & к \\ Ъ & D & F & H & j & I \end{matrix} \quad .$$

В этой конфигурации все пять соседей клетки F удалены не более чем на одну клетку от F, хотя некоторые из них и находятся в другой плоскости битов.

Для такого метода плиссирования правило SCARVES принимает вид

```
NEW-EXPERIMENT  N/VONN  &/CENTERS
                                    NOW
                            &CENTER
                                    4SUM0
 EAST CENTER' WEST'  WEST + + +
                                    4SUM1
EAST EAST' CENTER  WEST' + + +
                                    PLEATED-SCARVES
                          NOW  IF
4SUM0 2 = NORTH  XOR  >PLNO         \ прошлое только что
4SUH1 2 = NORTH' XOR  >PLN1 ELSE    \ вытеснено за северный
               SOUTHS >PLNA THEN    \ край
               CENTER >PLN2       ;
MAKE-TABLE  PLEATED-SCARVES
```

Так как массив был сжат в направлении запад-восток, то одна единица пространства является теперь половиной размера одной единицы времени, а световой конус стягивает более острый угол, как это видно из рис. 9.9Ь.

Глава 10

ТОЖДЕСТВЕННОСТЬ И ДВИЖЕНИЕ

> Фермер жаловался, что у него замерзли ноги: плед был слишком короткий. "Это легко поправить, - сказала его жена. - Я отрежу нужный кусок сверху и пришью его снизу!"

Конкурентный рост типа обсужденного в предыдущих главах может породить некоторый вид движения. Скажем, комок вещества размывается с одной стороны и наращивается за счет отложений с другой стороны: будет казаться, что он перемещается подобно амебе. В этом контексте тождественность и движение являются статистическими свойствами, которые исчезают, когда мы исследуем явление на уровне одиночной клетки.

Несколько правил, которые мы ввели, приводят не только к такому статистическому движению, но также и к более микроскопическому типу движения. Например, глайдеры из игры LIFE (разд. 3.1) являются структурами, которые на фоне нулей воспроизводятся правилом (после ряда шагов) в новой позиции, куда они смещаются после ряда шагов. Здесь налицо распознаваемый объект, который передвинулся. Правило SCARVES разд. 9.7 также поддерживает богатый спектр таких частиц.

Простейшей частицей была бы единица или нуль, движущиеся на однородном фоне нулей и единиц. Поэтому в этой главе наши методы рассмотрения тождественности и движения на микроскопическом уровне будут основываться на правилах, которые дают точное, детерминированное, микроскопическое сохранение единиц и нулей. После того, как мы освоимся с сохранением таких "материальных" объектов, нам будет легко обобщить наши методы на сохранение более абстрактных величин, таких, как энергия или импульс, и наиболее абстрактной из всех величин, а именно *информации*.

Для более простых экспериментов с сохранением материальных объектов мы будем использовать только уже введенные средства, которые CAM предоставляет для этой определенной цели и которые реализуют прямо в аппаратном виде некоторые из методов, применяемых в этих простых экспериментах.

10.1. Случайное блуждание

Чтобы "переместить" в клеточном автомате частицу из одного положения в другое, необходимо *создать ее копию* там и *удалить* ее отсюда. Несмотря на то что эти два действия выполняются в *различных* местах, они должны составлять две части *неделимой* операции - иначе частицы будут размножаться или исчезать - и, следовательно, должны быть тщательно скоординированы, как мы видели в разд. 9.1. Здесь мы рассмотрим наихудшую возможную ситуацию: решения двигаться или не двигаться, идти в одном или в другом направлении принимаются частицей мгновенно, так что точные схемы передвижения не могут быть известны заранее.

Небольшие частички чернил, взвешенные в воде, видны под микроскопом в состоянии непрерывного, нерегулярного движения - результат бесчисленных столкновений с молекулами воды в активном тепловом движении. В этом *броуновском движении* поведение отдельной частички чернил может быть аппроксимировано *случайным блужданием:* на каждом шаге частица передвигается на одну единицу расстояния вправо или влево[1] в зависимости от исхода бросания кости.

Поскольку мы имеем *только одну частицу,* то эту модель нетрудно реализовать на клеточном автомате. Например, в CAM мы можем использовать для частицы плоскость 0, а плоскость 1 как генератор случайных чисел, интерпретируя о в последней плоскости как "влево", а 1 как "вправо". В словесной форме команды для клетки имеют следующий вид:

> Если частица у вас, удалите ее: она будет подхваченд одним из ваших соседей.
>
> Если частица справа от вас и лежащий снизу случайный бит сообщает "влево", сделайте копию частицы.
>
> Аналогично, если частица слева от вас и случайный бит сообщает "вправо", сделайте копию частицы.

Результат показан на рис. 10.1а (где второе измерение показывает продвижение частицы во времени, как объяснено в разд. 9.7).

Ресурсы клеточного автомата тратятся неэффективно, если его использовать для одной частицы. Однако, когда мы хотим моделировать больше чем одну частицу, возникают трудности: что делать, когда две частицы, приходящие одна справа, а другая слева, пытаются ступить на один и тот же квадрат?

[1] Для простоты мы временно ограничиваем наше внимание одномерными моделями.

Предоставить отдельную плоскость битов для каждой частицы?[1] Модифицировать правило так, чтобы частицы как-нибудь отталкивали друг друга? Но тогда путь одиночной частицы не был бы идеальным случайным блужданием, так как он зависел бы от внешних по отношению к частице факторов.

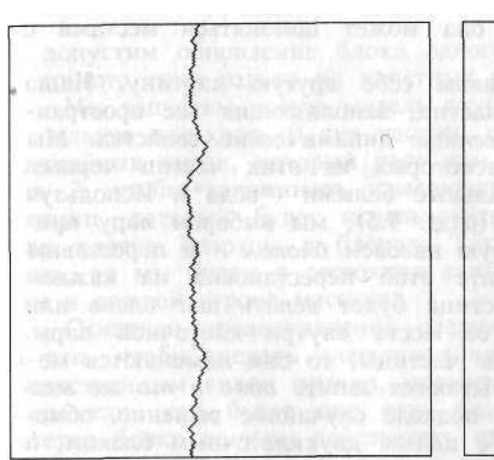
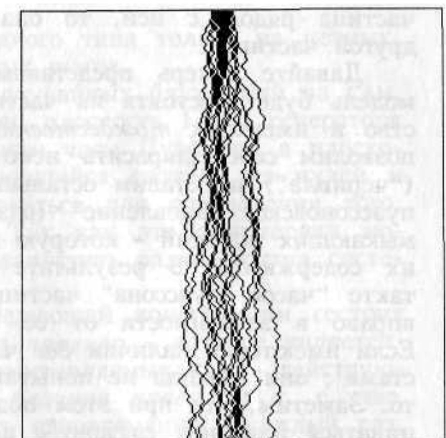



Рис. 10.1. Случайное блуждание: (а) пространственно-временная история одночастичной системы, использующей метод "влево/вправо"; (Ь) система многих частиц, использующая метод "замены".

Должно быть ясно, что это *концептуальная* трудность, а не трудность реализации.

> При одной плоскости битов на частицу, пути являются истинно случайными блужданиями. Однако произвольно большое число частиц может заканчивать траектории в одном и том же месте; это не наблюдается в природе, где молекулы в сущности непроницаемы друг для друга.
>
> При единственной плоскости битов кажется, что у нас нет иного выбора, кроме как направить частицу прочь от случайно выбранного ею пути, когда возникает спорная ситуация. Однако это испортило бы статистику *идеального случайного блуждания;* но на том уровне, который мы рассматриваем, именно *эта* статистика наблюдается в природе.

Ну, а как это делает *природа?*.

[1] В этом случае не может быть достигнута какая-либо экономия машинных ресурсов и модель более эффективно управляется обычным компьютером.

10.2. Случайные перемещения

В броуновском движении частицы перемещаются не в вакууме. В каждом месте мы имеем либо чернила, *либо воду,* которая также непроницаема. Когда частица передвигается, она *меняется* местами с некоторым количеством воды; если имеется частица рядом с ней, то она может поменяться местами с другой частицей.

Давайте теперь представим себе другую картину. Наша. модель будет состоять из частиц, заполняющих все пространство и имеющих *тождественные* динамические свойства. Мы позволим себе закрасить некоторых из этих частиц черным ("чернила") и оставим остальные белыми ("вода"). Используя пуассоновское обновление (разд. 9.5), мы выберем пару примыкающих позиций - которую назовем *блоком* - и переставим их содержимое. В результате этой перестановки на каждом такте "часов Пуассона" частица будет делать шаг влево или вправо в зависимости от ее места внутри клеточной пары. Если имеются в наличии *две* частицы, то они поменяются местами[1]; они никогда не попытаются занять *одно и то же* место. Заметим, что при этом подходе случайное решение, обмениваться или нет, связано с целым двухклеточным блоком, а не одиночной клеткой.

Правило перестановки является "нечувствительным к данным", т. е. обмен происходит одинаково, независимо от того, каково содержимое блока. Мы могли бы обновить массив, *даже не взглянув на него* Это гарантирует нам, что путь отдельной частицы будет иметь ту же статистику независимо от того, имеется одна частица или же целая компания!

Обновление по Пуассону гарантирует, что в любой момент мы будем пытаться заменить только один блок. Действительно, приведенный выше способ "перестановки" бессмыслен, если он применяется к двум перекрывающимся блокам:

$$\cdots\text{~}a\text{Ъ_}c\text{---} \qquad ; \tag{10.1}$$

т. е. содержимое клетки *b* не может быть заменено *одновременно* с заменой содержимого клеток а и с. С другой стороны, идеальное обновление по Пуассону может имитироваться синхронным клеточным автоматом только в рамках бесконечно медленного моделирования, как объяснено в разд. 9.5. Намного более быстрое моделирование может быть достигнуто введением подходящего порядка обновления, а именно мы разделим

[1] Если все частицы чернил имеют один и тот же цвет, то этот случай неотличим от случая полного отсутствия движения.

блоки на две группы - четную и нечетную, как показано ниже,

```
         чет.        чет.        чет.
... O    1    2    3    1    5    6 ... ,
            нечет.      нечет.      нечет.
```

и допустим обновление блока одного типа только на четных, а другого типа только на нечетных шагах.

Мы запустим новую модель случайных блужданий на CAM, используя плоскость 0 для частиц, плоскость 1 для генератора случайных чисел, который даст нам часы Пуассона, а плоскость 2, чтобы запоминать изменяющийся паттерн из нулей и единиц, который будет использоваться для определения того, как клетки спарены в блоках. Так как это одномерная модель, то мы будем в состоянии запустить разные копии системы в каждой строке массива.

Основное предназначение следующей конструкции состоит в том, чтобы сделать *клеточное правило* - которое является единственным видом правил, предоставляемых CAM - действующим так, как будто оно есть *блочное правило*[1]; т. е. две клетки блока должны действовать одновременно - каждый раз, когда они вообще действуют - и раздельно выполнять две части неделимой операции (в данном случае "перестановки"). Эта конструкция будет отправной точкой нашего обсуждения окрестности Марголуса в гл. 12.

Мы отдельно определим различные "части" нашего механизма и соберем их все вместе в конце.

Чередующаяся конфигурация блоков. Перед началом эксперимента запомним в каждой строке плоскости 2 конфигурацию

```
010101010101... ,
```

где о будет использоваться для определения левого элемента блока, а 1 правого элемента. На следующем шаге нам хотелось бы иметь конфигурацию

```
101010101010...
```

и т. д. в чередующемся порядке; это может быть учтено с помощью простой компоненты правила

```
(CAM-B)                       : CHANGE-GRID
          CENTER  NOT  >PLN2 ;
```

[1]В разд. 16.5 мы кратко исследуем второй подход к этой проблеме.

которая образует дополнение конфигурации. (Комментарий CAM-B является напоминанием о том, что мы работаем в среде CAM-в, так как задаем правило для плоскости 2.)

Чтобы мы могли запустить множество копий системы, строка чередующихся нулей и единиц была повторена на каждой строке экрана, так что плоскость 2 содержит чередующиеся вертикальные полоски нулей и единиц (см. разд. 14.3).

Маркеры "влево/вправо" будут использоваться CAM-A, где они будут видимы под именем соседа &CENTER; для ясности мы определим правило

```
(CAM-A)                          : LEFT/RITE (-- 0|1)
                        &CENTER ;
```

Мультиплексирование соседей. Независимо от того, в каком вида блоке мы находимся, соответствующий сосед клетки CENTER плоскости 0 является *противоположным* ей в блоке, т. е. EAST для левой клетки и WEST для правой. По этой причине будет удобно определить

```
(CAM-A)                          : OPPOSITE
        LEFT/RITE  { EAST  WEST } ;
                                 : OPPOSITE'
        LEFT/RITE  { EAST' WEST'} ;
```

так что, задав клетку как текущий 'CENTER' внимания, можно ссылаться на ее компаньона по блоку как 'OPPOSITE'.

Блочные часы. Плоскость 1 будет содержать обычный элементарный генератор случайных чисел (разд. 8.2), определенный правилом

```
(CAM-A)                          : STIR
       CENTER'
       NORTH'  WEST' SOUTH' EAST'
       AND  XOR  XOR  XOR  >PLN1 ;
```

На каждом шаге этот генератор обеспечит отличный для каждой клетки случайный исход. Чтобы обе клетки могли обнаружить один и тот же сигнал активации, мы обрабатываем два бита так, что результат одинаков для двух клеток:

```
(CAM-A)                          : POISSON-CLOCK (— 0|1)
           CENTER' OPPOSITE' XOR ;
```

Так как STIR дает равномерную смесь нулей и единиц, операция XOR двух случайных битов также даст равномерную

смесь, и значит, эти пуассоновские часы будут тикать в среднем один раз на каждые два шага.

И наконец, перестановка! В этом месте мы готовы написать основную компоненту правила, а именно компоненту для частиц в плоскости О

```
(CAM-A)                 : 1D-RANDOM-WALK
          POISSON-CLOCK IF    \ считать показания часов Пуассона
             OPPOSITE ELSE    \  и соответственно
               CENTER THEN    \  произвести перестановку
                    >PLNO ;   \  или оставить как было
```

Соединение всего вместе. Следующая запись является изложением законченного рецепта:

```
NEW-EXPERIMENT
CAM-B
        : CHANGE-LATTICE                      CENTER NOT >PLN2

CAM-A N/VONN «./CENTERS
         STIR                                            >PLN1
         LEFT/RITE                                «.CENTER
         OPPOSITE          LEFT/RITE { EAST  WEST  }
         OPPOSITE'         LEFT/RITE { EAST' WEST' }
         POISSON-CLOCK        CENTER' OPPOSITE' XOR
         1D-RANDOM-WALK                                  >PLNO

MAKE-TABLE  CHANGE-LATTICE
MAKE-TABLE  STIR
MAKE-TABLE  1D-RANDOM-WALK
```

Мы начнем эксперимент с заполнения в среднем трети каждой строки чернилами, а остальной части водой (рис. 10.2а): вода и чернила будут постоянно диффундировать друг в друга (Ь), что приводит в конечном счете к равномерной смеси (с).

Мы преуспели в создании микроскопической модели *диффузии,* которая замечательно реалистична и устойчива. Если мы определим *p* как *плотность* частиц чернил в относительно большой области массива, то эта величина будет непрерывно изменяться во времени и пространстве. В противоположность уравнению диффузии

$$\frac{dp}{It} = cPp \tag{10.2}$$

которое* является предельным случаем макроскопического приближения, наша модель отражает важные физические детали, такие, как непроницаемость тел и конечная скорость распространения информации[1]. С другой стороны, по мере того как мы изучаем ее во все большем масштабе, наша модель сводится к (10.2); это объясняет, почему уравнение диффузии является допустимым инструментом моделирования в подходящем макроскопическом контексте.

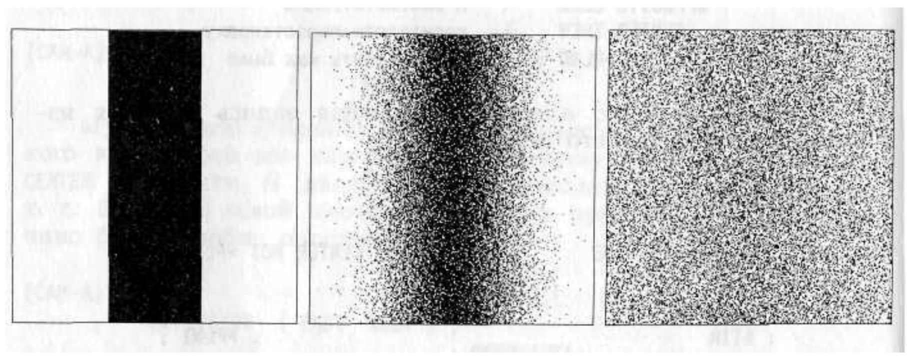


Рис. 10.2. Изначально разделенные (а), чернила и вода диффундируют друг в **друга (Ь), порождая в конечном счете однородную смесь (с).**

Если мы рассмотрим теперь единственную копию одномерной системы чернила/вода, то можно считать дисплей CAM пространственно-временной траекторией ее эволюции, как объяснено в разд. 9.7 (строка NOW проведена в плоскости 3). Если мы начнем этот эксперимент с плотного кластера частиц где-нибудь в середине строки NOW, ТО сможем увидеть, что каждая частица кластера следует отдельным зигзагообразным путем, как на рис. ЮЛЬ. Скорость диффузии составляет половину от скорости диффузии системы на рис. 10.1а, так как часы Пуассона тикают в среднем через один шаг.

Если мы остановим часы Пуассона во время эксперимента[2], то скорость диффузии, как и ожидалось, устремится к нулю (рис. 10.3а). Что произойдет, если мы заставим часы

[1] Уравнение диффузии предсказывает, что через сколь угодно малый промежуток времени *какая-то часть* чернил будет находиться на сколь угодно большом расстоянии от начального положения - что физически нереально.

[2] Например, заполнив всю плоскость 1 нулями.

неизменно тикать на каждом шаге[1]? (Конечно, такие синхронные часы вряд ли можно по-прежнему называть пуассоновскими.) Результат показан на рис. 10.3b; операции перестановки теперь вполне согласованы, и строка заменителей превращается в конвейер. Частицы летят вправо или влево с постоянной скоростью! Этот тип механизма переноса частиц будет использован в ряде моделей газа, которые вводятся позже.

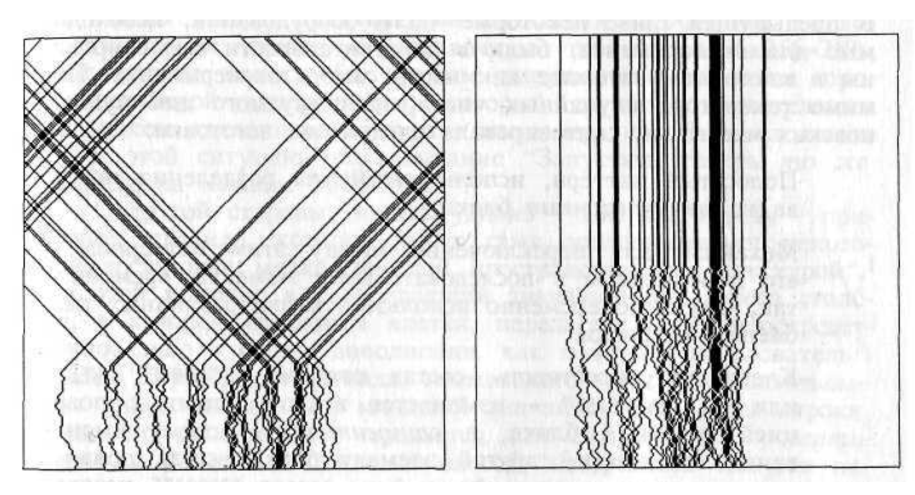


Рис. 10.3. (а) Когда пуассоновские часы останавливаются, частицы прекращают диффузию. Когда **часы тикают на каждом шаге, частица сохраняет движение всегда в одном и том же направлении - вправо или влево в зависимости от того, где она начала движение.**

[1] Заполнение плоскости 1 единицами не приведет к этому, так как STIR по-прежнему возвращает одни нули; заполнение ее чередующимися полосками нулей и единиц *даст* этот результат при данном определении **STIR.** Но это, конечно, ухищрения, и было бы лучше загрузить новое правило, где импульсы часов не ощущаются совсем.

Глава 11

ПСЕВДОСОСЕДИ

> Я всегда хотел иметь соседку точно такую же как ты! Я всегда хотел жить по соседству с тобой!
>
> [М-р Роджерс]

В предыдущей главе некоторые части оборудования, необходимые для эксперимента, были в действительности смоделированы в клеточном автомате как часть самого эксперимента. Помимо генератора случайных чисел, используемого для пуассоновских часов, мы синтезировали следующие заготовки:

> Полосатый паттерн, используемый для разделения массива на двухклеточные блоки.
>
> Механизм для переключения между этим паттерном и его дополнением в последовательные моменты времени - так, чтобы попеременно использовать два различных разбиения на блоки.
>
> Клеточную окрестность, состав которой - "CENTER[1], WEST" или "CENTER, EAST" - изменяется в соответствии с позицией клетки в блоке, и *относительную* схему именования, по которой другой элемент блока всегда появляется под одним и тем же именем соседа OPPOSITE независимо от того, оказывается ли он в *абсолютном* смысле западным или восточным соседом выбранной клетки.[1]

Так как они полезны в очень многих ситуациях, то CAM предоставляет различные средства этого типа прямо в *аппаратном виде;* таким образом плоскости битов и справочные таблицы могут быть лучше использованы, а само программирование упрощается. Пространственная и временная информация, полезная для построения разбиений на блоки и для других целей, доступна в форме *псевдососедей,* т. е. сигналов, которые могут быть непосредственно считаны справочной таблицей, но источником которых не является содержимое клетки. В следующей главе мы увидим, что некоторые из этих сигналов доступны также *косвенно,* в форме, которая для определенных целей даже более эффективна благодаря использованию соседей окрестности Марголуса.

[1] Справочная таблица по-прежнему физически присоединена к обеим линиям соседей, но в результате получается мультиплексирование этих двух линий в одну "виртуальную", несущую нужную информацию в нужный момент.

11.1. Пространственные фазы

Положение каждой клетки в массиве размером 256x256, который составляет клеточный автомат CAM, однозначно определяется ее адресом в строке и столбце (мы предположим, что и строки, и столбцы занумерованы от 0 до 256). Если бы правило могло полностью считать этот адрес, то оно могло бы определить разное поведение для каждой клетки ("Если ваш адрес <3,7>, делайте это, если <0,95>, делайте то, и т.д."). Не говоря уже о практической сложности реализации 16 дополнительных входов в справочную таблицу ($256x256=2^{16}$), этот подход явился бы грубым нарушением принципа пространственной однородности, который служит отличительным признаком клеточного автомата; например, какой смысл имело бы в этой ситуации высказывание "Запустить теперь *то же* правило на массиве 1000x1000"?

С другой стороны, иногда удобно - как мы видели - применять правила, которые могут сами воспользоваться некоторой достаточно мелкозернистой пространственной "текстурой".[1] По этой причине в CAM младший значащий бит адреса столбца, в котором находится клетка, передается (либо непосредственно, либо в форме дополнения, как пожелает пользователь^)) в виде внутреннего сигнала, называемого HORZ – *горизонтальная фаза;* аналогично, младший значащий бит адреса строки (или его дополнение) передается как сигнал VERT – *вертикальная фаза.* Единственный способ "настроиться" на эти передачи состоит в том, чтобы выбрать вспомогательную окрестность &HV (см. разд. 7.3.2), где они появляются соответственно как &HORZ и &VERT. Следовательно, &HORZ будет попеременно принимать значения о и 1 при продвижении вдоль строки массива; так как все строки синфазны, то это дает паттерн вертикальных полосок:

…010101010101…
…010101010101… ;
…010101010101…

аналогично, &VERT будет чередоваться вдоль *столбца,* давая горизонтальные полоски:

…000000000000…
..ЛИПШИЦ…
…000000000000…

[1] Кристалл NaCl является все же *однородным,* несмотря на чередование атомов Na и C1 в каждой строке; чтобы рассматривать его как однородный, следует учесть, что расположение атомов периодично.

[2] Смотрите разд. 11.2.

Применяя операцию XOR к соседям &HORZ и &VERT, получаем шахматный узор

```
...0101010101010...
...1010101010101...    ;
...0101010101010...
```

с другой стороны, объединенный вариант двух этих соседей - Ш, который является переменной с четырьмя состояниями, дает периодический паттерн

```
...0101010101010...
...2323232323232...
...0101010101010...      ,      где блок  01
...2323232323232...                       23
```

неограниченно повторяется как по горизонтали, так и по вертикали.

11.2. Временные фазы и фазовое управление

Хотя при написании правила удобно представлять себе слово соседства, скажем NORTH, возвращающим текущее состояние определенной клетки, важно помнить, что эти слова используются только программным обеспечением компьютера-хозяина при *компилировании* таблицы; их роль исчерпана, как только эта таблица передана CAM. Следовательно, слово NORTH не отвечает на вопрос, относящийся к реальному времени, "Каково текущее состояние в CAM северного соседа этой клетки?" - его значение имеет смысл только во время генерации таблицы.

Когда CAM работает, входные линии таблицы присоединены к задействованным аппаратным сигналам, которые находятся во взаимно-однозначном соответствии со словами соседства, используемыми при компиляции таблицы. Сигналы, соответствующие обычным соседям, таким как CENTER, NORTH, и т.д., поступают от плоскостей битов; кто подает сигналы, соответствующие псевдососедям?

Как упомянуто в предыдущем разделе, сигналы, соответствующие пространственным фазам &HORZ и &VERT, генерируются внутри системы горизонтальными и вертикальными счетчиками CAM, которые знают адреса каждой клетки в строке и столбце. Остальные фазовые сигналы, соответствующие псевдососедям &PHASE и &PHASE, упомянутым в разд. 7.3.2 (а также PHASE и PHASE, которые вводятся в разд. 12.5), управляются непосредственно компьютером-хозяином, который перед каждым шагом может независимо устанавливать каждый из них либо в о, ли-

бо в 1. Если сигнал, соответствующий &PHASE, установлен, скажем, в 1, то во время самого шага *всем* клеткам будет видно, что сосед &PHASE имеет значение 1, и следовательно, будет выглядеть при моделировании как пространственно-независимый (но, возможно, зависящий от времени[1]) параметр.

Определение правил, которые обращаются к слову псевдососедства, такому как &PHASE, конечно, бессмысленно, если соответствующий аппаратный сигнал не управляется компьютером-хозяином шаг за шагом при помощи необходимой последовательности значений *во время* самого эксперимента. Поэтому наряду с термином соседства PHASE существует команда вида

<значение> IS <&PHASE> ,

используемая для помещения определенных значений на соответствующую сигнальную линию. Подходящие для задания команд этого типа ситуации обсуждаются в оставшейся части этой главы; для справки мы приведем здесь полезные варианты и набор возможных значений аргументов для каждого из них.

ДИАПАЗОН	КОМАНДА	ПРИМЕЧАНИЯ
0 1	IS <&PHASE>	сигнал, видимый &PHASE
0 1	IS <&PHASE'>	сигнал, видимый &PHASE'
0 1 2 3	IS <&PHASES>	объединение этих назначений
0 1	IS <ORG-H>	источник сигнала HORZ
0 1	IS <0R6-V>	источник сигнала VERT
0 1 2 3	IS <ORG-HV>	объединение этих назначений
0 1	IS <PHASE>	сигнал, видимый PHASE
0 1	IS <PHASE'>	сигнал, видимый PHASE'
0 1 2 3	IS <PHASES>	объединение этих назначений

(11.1)

Сигнал пространственной фазы, такой как HORZ, конечно, принимает на одном и том же шаге разные значения в разных местах массива; т. е. для каждой клетки это значение зависит от *позиции* самой клетки, которая фиксирована, а не от прихоти экспериментатора в некоторый *момент*. Тем не менее, на каждом шаге экспериментатор может выбрать для HORZ либо паттерн 010101... , который начинается с 0 в *начале* масси-

[1] Мы, конечно, имеем в виду *время клеточного автомата,* которое идет вперед дискретными шагами. Тот факт, что во внутренней реализации САМ клетки заменяются последовательно, здесь не имеет значения: это внутреннее время скрыто от взгляда, как о&ьяснено в начале гл. 7.

ва, либо дополнительный к нему паттерн 101010..., который начинается с 1. Выбор значения в начале является, следовательно, *зависимой от времени* переменной, представленной в приведенной выше табл. (11.1) как <ORG-H>, ее значение по умолчанию 0. Аналогичные соображения применяются и к VERT. Два эти сигнала могут быть установлены вместе присвоением значения между о и з переменной <ORG-HV> (как показано выше). Упорядочение двух битов внутри этой общей переменной осуществляет равенство <ORG-HV> = <ORG-H> + 2x<ORG-V> .

11.3. Двухфазное правило

В гл. 9 мы видели, как подходящее правило может заставить *состояние клетки* проходить через многофазный цикл. Как иллюстрацию использования псевдососедей, здесь мы опишем простой эксперимент, где *само правило* циклически изменяется от шага к шагу.

Мы хотим синтезировать "составное" правило, называемое BORDER/HOLLOW, которое воздействует на плоскость битов следующим образом:

> На четных шагах наращивает границу из единиц вокруг любой клетки, которая находится в состоянии 1.
>
> На нечетных шагах опустошает любые сплошные области обращением в о всякой клетки, которая полностью окружена единицами.

Чтобы узнать, является ли время "четным" или "нечетным" правило наблюдает за псевдососедом &PHASE; мы сделаем так, что этот параметр будет принимать чередующиеся значения о и 1 на последовательных шагах при проведении эксперимента. Весь эксперимент строится следующим образом.

1. Мы начнем новую программу высказыванием

   ```
   NEW-EXPERIMENT
   ```

   Это устанавливает всю машину - аппаратную и программную части - в точно определенное состояние, устанавливаемое по умолчанию. Опции останутся бездействующими до тех пор, пока не будут явно приведены в действие.

2. Затем мы сделаем соответствующее объявление окрестности и отдельно запишем два "куска" правила

```
N/MOORE &/PHASES   \ в этой дополнительной окрестности'
                   \ &PHASE и &PHASE' видимы

                                  : BORDER (—новое состояние)
  CENTER NORTH SOUTH WEST EAST
  N.WEST N. EAST  S.WEST S.EAST
       OR OR OR OR OR OR OR OR ;
                                  : HOLLOW (-- новое состояние)
         NORTH SOUTH WEST EAST
  N.WEST N.EAST S.WEST S.EAST
  AND AND AND AND AND AND AND
                    IF 0 ELSE
                  CENTER THEN ;
```

3. Из двух этих частей мы сформируем одно фазочувствительное правило и загрузим его в справочную таблицу, передав его как аргумент правилу MAKE-TABLE :

```
                                    : BORDER/HOLLOW
  &PHASE { BORDER HOLLOW } >PLNO ;

  MAKE-TABLE BORDER/HOLLOW
```

4. В этом месте мы определяем дескриптор рабочего цикла, т. е. слово языка Forth, которое определяет, в какое значение должен быть установлен сигнал, соответствующий &PHASE, перед каждым шагом. Это слово, которое мы назвали EVEN/ODD, передается как аргумент команде MAKE-CYCLE (это весьма похоже на передачу дескриптора таблицы в качестве аргумента команде MAKE-TABLE). Возможные составляющие рабочего цикла обсуждаются в разд. 11.5; содержание правила здесь простое:

```
                                    : EVEN/ODD
                 0 IS <&PHASE>  STEP
                 1 IS <&PHASE>  STEP  ;

  MAKE-CYCLE  EVEN/ODD
```

Теперь мы готовы к работе. Клеточный автомат будет чередовать один шаг правила BORDER с одним шагом правила HOLLOW. В результате получим, в сущности, многократные пространственные производные; а именно, первый двухшаговый цикл будет извлекать контур всякой сплошной области единиц, имеющейся в исходной конфигурации, второй цикл будет вычерчивать контур контуров и т.д., заполняя в конечном счете все пространство сложными движущимися узорами (рис. 11.1)

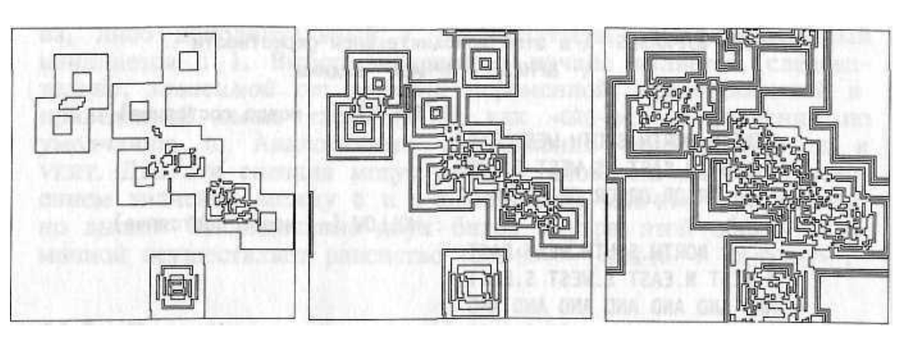


Рис. 11.1. Последовательные стадии правила BORDER/HOLLOW.

11.4, Инкрементное управление фазой

В предыдущем примере значение <&PHASE> было явно установлено в о или 1 на чередующихся шагах рабочего цикла. Так как фазовые переменные могут как читаться, так и записываться, то вместо определения нового значения фазы в *абсолютном* смысле можно было бы определить ее в *относительном* смысле, т. е. по отношению к значению фазы на предыдущем шаге.

Например, для двузначной переменной вроде <&PHASE> можно просто образовывать *дополнение* ее значения на каждом шаге. (При таком инкрементном подходе инициализация фазовой переменной в начале работы, если необходимо, должна быть обеспечена отдельной командой). Рабочий цикл должен быть установлен следующим образом:

```
: INIT-PHASE
        O IS <&PHASE> ;
: CHANGE-PHASE
        <&PHASE> NOT IS <&PHASE> ;
: EVEN/ODD
        STEP CHANGE-PHASE

MAKE-CYCLE EVEN/ODD
INIT-PHASE
```

Заметим, что команды вроде INIT-PHASE и CHANGE-PHASE, определенные выше отдельными подпрограммами, могут быть интерпретированы в произвольный момент в процессе моделирования[1]

[1] Например, вводом их с клавиатуры или связав их с какими-нибудь ее клавишами.

с целью изменить регулярное чередование значений фазы, в противном случае определенное рабочим циклом. Это один из способов достигнуть обращения времени в правилах, в которых направление времени управляется фазовой переменной (см. гл. 12 и 14).

11.5. Рабочий цикл

Если бы мы работали с машиной клеточных автоматов, в которой единственными относящимися к делу данными было содержание плоскостей битов и справочной таблицы, то, чтобы выполнить эксперимент достаточно было бы сказать "Вот правило, а вот начальная конфигурация - действуй!" Действительно, как показано в предыдущих главах, многие простые эксперименты могут быть выполнены на CAM именно в этом режиме.

С другой стороны, в составном правиле, таком как правило из разд. 11.3, полный цикл может состоять из нескольких шагов, каждый из которых выполняется в другой среде. Многие из средств CAM можно с пользой заставить исполнять разные роли в разных точках цикла: помимо управления фазами, может быть необходимо изменить цветовую карту, работу счетчика событий или даже заменить одну окрестность на другую.

Здесь мы обсудим основной итерационный цикл, по которому работает клеточный автомат на CAM. Он состоит из каркаса, к которому могут прикрепляться различные типы "декораций" с целью создать определенную среду рабочего цикла.

Команда STEP велит CAM выполнить шаг. Какой вид действия будет предпринят на этом шаге, определяется содержимым справочных таблиц и состоянием других сигналов, таких как псевдососеди; в совокупности они составляют *динамические параметры* машины; эти параметры могут быть модифицированы между двумя последовательными шагами.[1] Помимо использования этих параметров как "входных", сам шаг может воз-

[1] Между двумя последовательными шагами возможна также модификация содержимого плоскостей битов, представляющего собой *переменные состояния* системы. Аппаратная часть CAM предоставляет между шагами короткое временное окно, в пределах которого компьютер-хозяин может "обращаться" к CAM без нарушения выполнения шагов. Времени в этом окне достаточно, чтобы обновить все динамические параметры, которых немного; за небольшое оставшееся время можно считать или изменить на ходу несколько битов плоскости или справочной таблицы. Более обширные манипуляции превысили бы ширину окна и заставили бы CAM приостановить моделирование на один или более шагов. За счет замедления моделирования, данные плоскости битов и справочной таблицы могут быть по желанию модифицированы между шагами.

вращать выходные данные - например, значение счетчика событий.

Таким образом, для того чтобы выполнить эксперимент, нужно (а) сначала установить все относящиеся к делу динамические параметры; (Ь) определить, какие изменения (если имеются) должны быть сделаны в некоторых параметрах перед каждым шагом[2] и (с) определить, какие данные (если имеются) должны контролироваться или собираться после каждого шага.

В сущности, "распорядок дня" эксперимента может быть записан следующим образом:

```
: AGENDA
<служебные операции>
        STEP
<служебные операции>
        STEP
<служебные операции>

<служебные операции>
        STEP
<служебные операции> ;
```

(повторяющиеся части, конечно, могли бы быть включены в операторы цикла), и действительно этот режим работы САМ полезен для заготовленных заранее демонстрационных примеров и других специализированных применений.

Если мы запустим правило AGENDA, то машина будет работать на полной скорости (если сами <служебные операции> не дают задержку); если мы хотим приостановить моделирование, нам придется прекратить выполнение программы - при этом будет сложно узнать, где мы находились и как правильно ее продолжить. Мы, конечно, могли бы разместить по всей программе явные указания считывать коды с управляющей панели, осуществлять синхронизацию с внешними часами, останавливаться в определенных местах, и т. д. Однако все эти команды *контроля времени* обычно лучше отделить от основных вопросов моделирования - а именно, *что* делать и *в каком порядке*.

Это разделение между контролем времени и установлением последовательности действий достигается в САМ выделением программы AGENDA в качестве *сопрограммы* основной управляющей программы, причем последняя будет в любом желаемом месте вызывать сопрограмму. Сопрограмма, используемая та-

[2] **Когда несколько модулей САМ используются вместе, параметры для каждого из них могут быть независимо модифицированы между шагами.**

ким образом, будет называться *рабочим циклом*. Основная программа будет передавать управление рабочему циклу, когда захочет выполнить следующий шаг; рабочий цикл будет обратно передавать управление основной программе, когда начат данный шаг и установлены параметры для следующего. Как основная программа, так и рабочий цикл всегда продолжают с того места, где они остановились, когда им передали управление.

Грамматика является следующей. Мы сообщаем САМ, ЧТО слово AGENDA (например) должно быть текущим рабочим циклом, объявляя

```
MAKE-CYCLE AGENDA
```

В рабочем цикле вхождения слова STEP имеют особое значение; как только слово AGENDA объявлено в качестве текущего рабочего цикла, оно начнет выполнение этой программы и будет продолжать его до тех пор, пока не обнаружится первое вхождение STEP. Как раз перед выполнением этого слова AGENDA уснет: все машинные параметры установлены для выполнения первого шага, но сам шаг еще не был выполнен.

Основная программа будет отвечать за генерацию "тиканья" моделирующих часов. Тактовые импульсы могут следовать быстро или медленно, они могут быть приостановлены, возобновлены, посланы группами по два и т.д., как угодно. На каждом "тике" запускается команда; эта команда будит программу AGENDA, которая выполняет отложенную команду STEP, и сдерживает ее выполнение до тех пор, пока не находит следующее вхождение слова STEP - в этом месте она засыпает снова. Теперь первый шаг выполнен и машина установлена для следующего, и т. д.

Когда программа AGENDA достигает конца ее кода, она возвращается к началу: чтобы повторно запустить цикл, нет необходимости в явном операторе цикла.[1]

Например, для всех клеточных автоматов, введенных до сих пор,[2] было бы достаточно сказать

```
MAKE-CYCLE STEP
```

Таким образом, поскольку никакие параметры не изменяются между шагами, задание, которое должно быть выполнено на каждом такте часов, состоит только из команды STEP.

[1] Команда FINISH-CYCLE будет продолжать выполнение текущего цикла до конца его кода, а затем укладывает его спать. Это полезно, если во время интерактивного сеанса необходимо упорядоченным образом переключиться с заданного рабочего цикла на другой.

[2] Исключая, конечно, специальный пример из разд. 11.3.

Лишенное окружения слово STEP является рабочим циклом по умолчанию; поэтому в приложениях, не требующих пользовательского рабочего цикла, не нужно даже знать о существовании этого механизма составления расписаний.

11.6. Чередующиеся пространственные текстуры

Текстуры, которые могут быть созданы в одном или двух измерениях с помощью пространственных фаз (такие как разбиения на блоки, полосы или шахматные доски) особенно полезны, когда регулярное чередование фазовых значений добавляется к *временному* измерению системы.

Один пример этого дает эксперимент со случайным блужданием разд. 10.1, где мы использовали плоскость битов 2, чтобы подготовить паттерн вертикальных полос, от которого на каждом шаге формировалось дополнение. Чтобы достигнуть того же результата, используя псевдососедей с пространственно-зависимой фазой, можно выбрать вспомогательную окрестность &HV (а не &/CENTERS) и применить следующее определение маркера блока LEFT/RITE, используемое в таком примере:

```
                        : LEFT/RITE
                  &HORZ ;
```

Подходящим рабочим циклом (см. табл. (11.1) и разд. 11.4) был бы

```
                        : ALT-STRIPES
  STEP <ORG-H> NOT IS <ORG-H> ;
```

Если важно начать цикл с определенного выбора паттерна, то рабочий цикл можно определить следующим образом (см. разд. 11.4):

```
                        : ALT-STRIPES
         0 IS <ORG-H> STEP
         1 IS <ORG-H> STEP ;
```

Заметим, что для получения доступа к линиям связи псевдососедей окрестности &HV нужно рассоединить две линии соседей окрестности &CENTERS. В следующем разделе мы покажем, как можно "иметь пирог и одновременно есть его".

В любом случае, хотя эти конкретные компромиссы являются специфическими для САМ, ОНИ отражают реальную кон-

куренцию за ресурсы и, следовательно, иллюстрируют компромиссы, которые придется рассматривать и в других высокоэффективных моделирующих средах.

Стандартный метод имитации решеточных моделей физических систем состоит в том, чтобы обновлять на одном шаге клетки, располагающиеся на "черной" подрешетке шахматной доски, а на следующем те, которые принадлежат "белой" подрешетке (см. разд. 17.3). Интуитивные предпосылки такого подхода даются примерами разд. 9.5 и 10.1; на самом деле два подхода, основанные на "чередующихся разбиениях" и "чередующихся подрешетках", как объяснено в разд. 17.7, являются в некотором смысле двойственными аспектами одного и того же подхода.

Глава 12

ОКРЕСТНОСТЬ **МАРГОЛУСА**

Делай для других так же, как ты желаешь, чтобы другие делали для тебя.

[Золотое правило]

В данной главе мы обсудим средства поддержки, предоставляемые CAM для понятия *окрестности Марголуса,* т. е. схемы взаимосвязи клеток, полезной в физическом моделировании, особенно в связи с микроскопической обратимостью. Концептуальное обоснование окрестности Марголуса и характерные ее применения будут даны в части III; здесь же мы коснемся в основном функциональных аспектов окрестности Марголуса: что это такое и как это использовать.

12.1. Правила для блока

Поступая как в разд. 10.2, определим новый вид клеточного автомата - *клеточный автомат на разбиении:*[1]

1. Массив клеток разбит на множество конечных, отдельных однородных частей - *блоков.*

2. Дается *правило для блока,* рассматривающее и обновляющее содержимое всего блока (а не отдельной клетки как в обычном клеточном автомате); пример приведен в табл. (12.2). Одно и то же правило применяется ко всем блокам. Заметим, что блоки не перекрываются и нет обмена информацией между соседними блоками.

3. Разбиение меняют от шага к шагу так, чтобы было некоторое перекрытие блоков, используемых на соседних шагах.

Пункт 3 существен: если бы мы использовали одно и то же разбиение на каждом шаге, то клеточный автомат оказался бы разбит на совокупность независимых подсистем.

[1] Мы не меняем правил в ходе игры! Используя большее количество состояний и соседей, систему данного вида всегда можно преобразовать в обычный клеточный автомат.

В этой книге мы будем использовать только простейшую схему разбиения, а именно:

> массив клеток разбит на блоки размером 2x2;
>
> шаги, в которых блоки соответствуют *четной* решетке на "бумаге в клеточку", чередуются с шагами, использующими *нечетную* решетку, как показано на рис. 12.1.

Такую схему разбиения называют *окрестностью Марголуса,* и она непосредственно поддерживается САМ.

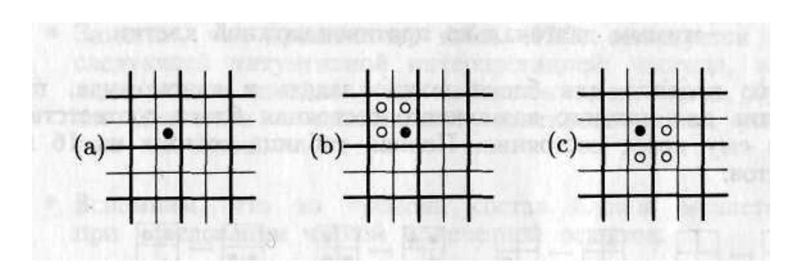


Рис. 12.1. Блоки 2x2 окрестности Марголуса; на последовательных шагах чередуются четная (жирные линии) и нечетная решетки (тонкие линии). В зависимости от того, какая решетка используется, клетка, помеченная в (а), будет иметь окрестностью либо четный блок (Ь), либо нечетный блок (с).

Заметим, что окрестность Марголуса использует только *два* разбиения (четная решетка и нечетная решетка). В общем случае ясно, что различных разбиений, используемых шаг за шагом правилом клеточного автомата на разбиении, должно быть *ограниченное* число и они должны циклически повторяться, чтобы сохранялась *однородность* пространства и времени.[1]

[1] Клеточный автомат можно считать динамической системой с *периодическими* в пространстве и времени законами. В обычном клеточном автомате один временной цикл соответствует одному шагу правила, и вдоль каждой из двух осей один пространственный цикл соответствует одной клетке. Клеточный автомат на разбиении, хотя и определен на той же самой пространственной решетке, обладает законами, период которых *больше*. А именно, чтобы завершить один цикл вдоль временной оси, для окрестности Марголуса требуется два шага (один шаг использует четную решетку, другой - нечетную) и требуется две клетки (длина блока 2x2), чтобы охватить цикл вдоль любой из пространственных осей.

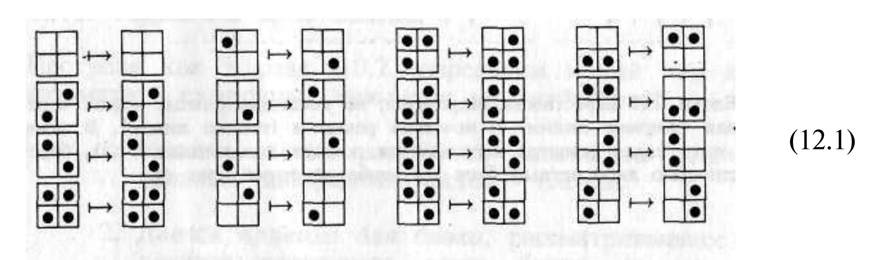
12.2. Частицы в движении

Чтобы привести простой пример использования окрестности Марголуса, построим модель стилизованного газа, состоящего из частиц, движущихся с одинаковыми скоростями *без взаимодействий*. Клетка может находиться в одном из двух возможных состояний: может быть пустой (" ") или содержать в себе частицу ("•").

Правило состоит в следующем:

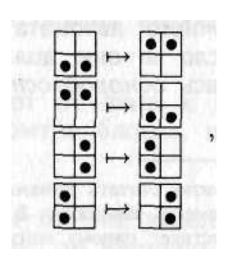
* SWAP-ON-DIAG В каждом блоке 2x2 данного разбиения поменять местами содержимое каждой клетки с содержимым диагонально противоположной клетки.

Такое правило для блока можно задать в явном виде, перечислив для каждого возможного состояния блока соответствующее ему новое состояние. Полная таблица состоит из 16 элементов:

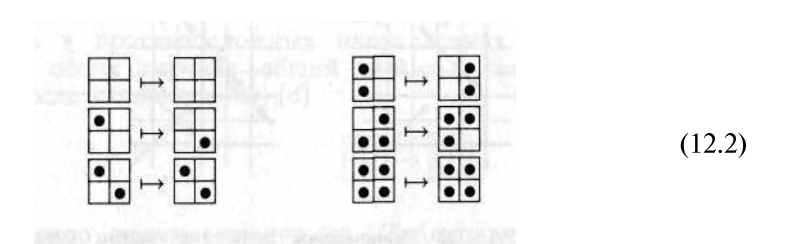


(12.1)

Так как данное правило инвариантно относительно поворота, то некоторые элементы табл. (12.1), такие, например, как



совпадают с точностью до поворота; для простоты и краткости будет удобно использовать сокращенную форму табл. (12.1), в которой каждый класс эквивалентности относительно поворота представлен единственным элементом, поэтому только шесть, а не шестнадцать элементов необходимы в таблице:



(12.2)

Поучительно детально проанализировать это правило.

- Заметим, что предписание табл. (12.2) согласуется со следующей интуитивной интерпретацией: частица, находящаяся в данном углу занимаемого ею блока, будет двигаться в диагонально противоположный угол того же блока.

* Вспомним, что во времени состав блоков меняется при чередовании четной и нечетной решеток.

* Рассмотрим изолированную частицу. Если для первого шага выбрать, скажем, четную (жирные линии) решетку и взять частицу, лежащую в левом верхнем углу блока (рис. 12.2а), то данное правило заставляет частицу двигаться в правый нижний угол, т. е. совершать движение в направлении вниз и вправо.

- На следующем шаге используем нечетную решетку. В этом случае данная частица *снова* окажется в левом верхнем углу блока, и рассматриваемое правило снова вовлечет ее в движение вниз и вправо.

- Продолжим в том же духе, чередуя разбиения. Каждая частица будет двигаться по диагонали с одинаковой скоростью, а в каком из четырех возможных направлений - зависит от ее начального положения.

* Когда две частицы сталкиваются лоб в лоб, они меняются местами на диагонали в соответствии с сутью правила (SWAP-ON-DIAG меняет местами все что угодно - частицы или вакуум - даже не интересуясь этим) и продолжают свое движение. Частицы, движущиеся по пересекающимся диагоналям, продолжают свое движение, не замечая друг друга (рис.12.2Ь).

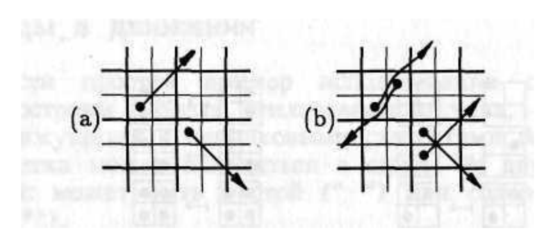


Рис. 12.2. SWAP-ON-DIAG. (а) Равномерное движение частиц (здесь и далее частицы показаны в начале четного шага - при этом рассматриваемые блоки выделены жирными линиями); (Ь) Частицы проходят одна "через" другую без взаимодействия.

Итоговая динамика весьма тривиальна: поскольку частицы не взаимодействуют друг с другом, то каждая частица является изолированной системой. Однако представленный здесь механизм переноса частицы - главная составная часть в ряде правил (которые будут введены начиная со следующего раздела), представляющих интерес для вычислений и физического моделирования.

МЛ, Столкновения

Обеспечив простой механизм равномерного движения, мы сейчас дополним его механизмом столкновения, в котором аналогия с физикой будет по возможности сохранена.

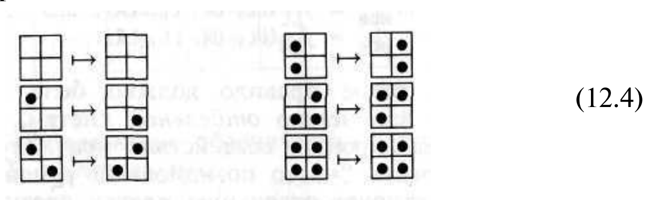
В стилизованном газе, описанном в предыдущем разделе, частицы можно считать имеющими *массу,* а, следовательно, и *кинетическую энергию.* Поскольку все частицы идентичны и движутся с одинаковой скоростью, то все они обладают одинаковой энергией; таким образом, сохранение энергии эквивалентно сохранению частиц, и, конечно, частицы сохраняются правилом "перестановки" SWAP-ON-DIAG. Поскольку частицы движутся равномерно, то тривиально сохраняется и импульс. Анализ справочной таблицы (12.2) показывает, что если необходимо ввести столкновения, сохраняющие и энергию, и импульс, то для этого имеется единственная возможность.[1] А именно, тре-/ тий элемент таблицы, где две частицы, движущиеся по диаго-

[1] Заметим, что первый и последний элементы (12.2) не могут быть изменены вследствие сохранения энергии. Второй элемент, описывающий свободное движение частиц, характеризует рассматриваемое нами семейство газов. Замена этого элемента (которая возможна лишь ценой нарушения еще одной симметрии на микроскопическом уровне) приведет к другому семейству газов, обсуждаемому в разд. 12.7

иали в разных направлениях, сталкиваются лоб в лоб, может быть изменен так, чтобы частицы, вопреки обыкновению, разбежались в противоположных направлениях по *другой* диагонали (в обоих случаях общий импульс равен нулю как до, так и после столкновения):

(12.3)

Это правило называется HPP-GAS. Таблица для него выглядит следующим образом:



(12.4)

Рис. 12.3 иллюстрирует сущность этого правила.

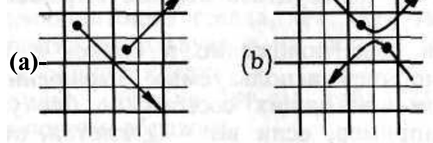


Рис. 12.3. HPP-GAS. (а) Равномерное движение изолированной частицы. (Ь) Частицы после лобового столкновения разлетаются вдоль другой диагонали.

Здесь уместно следующее замечание. Обозначим через *а* и *Ь* частицы, испытывающие столкновение, и предположим, что все частицы идентичны. Вследствие симметрии исхода, не имеет смысла вопрос, какая из двух выходящих из столкновения частиц *а* и какая *Ъ*. При желании можно считать, что столкновение уничтожает исходную пару и создает новую.

Несмотря на то, что положения частиц ограничены дискретным набором позиций и что движение происходит с постоянной скоростью в одном из четырех направлений, *макроскопические* свойства газа HPP-GAS идентичны свойствам физических газов. Этот важный результат будет обсужден в гл. 16, которая посвящена динамике жидкостей.

12.4. Как преобразовать правило для блока в правило для клетки

Правило для блока, как указано в табл. (12.2), определяет новое состояние каждой клетки данного блока (обозначим их через UL, UR, LL, LR для левой верхней, правой верхней, левой нижней, правой нижней) как функцию текущего состояния тех же четырех клеток по формулам

HOB AJL L > UR > LL > LR > '
UL HOB " /UR (UL > UR > LL > LR >>
UL HOB = /IL (uL' UR > LL > LR)'
UL HO B - /LR (UL > UR > LL' LR > -

В CAM, однако, такое правило должно быть представлено в виде рецепта по изменению *отдельной* клетки, а не всего блока; чтобы промоделировать воздействия правила для блока, мы должны использовать "чисто позиционный рецепт", который по существу инструктирует отдельную клетку следующим образом:

1. Если в текущей решетке вы оказались UL-клеткой блока, то пользуйтесь первым выражением в 12.5; если вы - UR-клетка, то пользуйтесь вторым выражением, и т.д.

2. Определив свое положение в блоке, а следовательно и одно из четырех используемых выражений, нужно выяснить положение ваших соседей по блоку по отношению к вам. Например, если вы - UL-клетка, то аргумент UR в этом выражении совпадает с вашим соседом EAST, LL-клетка - с вашим соседом SOUTH, и LR-клетка - с вашим соседом s.EAST; а поскольку вы - UL-клетка, то аргумент UL - не что иное, как вы сами, т. е. сосед по имени CENTER.

   Другими словами, первое выражение в 12.5 применимо *только* для UL-клеток, и для них оно эквивалентно

$\text{CENTER}_{\text{HOB}}$ «• $/_{\text{UL}}$(CENTER, EAST, SOUTH, S.EAST).

Все это выглядит достаточно сложно, однако положение существенно упрощается специальными аппаратными возможностями и соответствующим программным обеспечением. Объявление *окрестности Марголуса* (в любой из своих разновидностей N/MARG, N/MARG-HV и N/MAR6-PH) активирует среди прочего ряд слов соседства особенно удобных для правил, инвариантных по отношению к повороту.[1] По отношению к любой из

[1] Смотри в разд. 12.5 полный список соседей в смысле Марголуса.

четырех клеток, рассматриваемых как текущая CENTER (которая может находиться в любом из четырех положений в блоке), остальные три клетки блока назовем cw, CCW и OPP (из вашего угла блока три других угла могут быть достигнуты движением *по часовой стрелке* (ClockWize), *против часовой стрелки* (Counter-ClockWize), или *диагонально противоположным* (OPPosite), как показано на рисунке (звездочка указывает клетку CENTER).

UL	UR
LL	LR

*	CW	CCW	*	CW	OPP	OPP	CCW
CCW	OPP	OPP	CW	*	CCW	CW	*

(12.6)

С этими *относительными* обозначениями правило SWAP-ON-DIAG таблицы (12.2) выглядит так:

```
                    : SWAP-ON-DIAG
        OPP >PLNO ;
```

т. е. новое состояние любой клетки является текущим состоянием ее противоположного соседа, OPP. Заметим, что OPP "знает", какая решетка используется на данном шаге и где в блоке расположена данная клетка.

Аналогично, правило HPP-GAS табл. (12.4) может быть выражено следующим образом:

```
                         COLLISION  (- f|t)
          CENTER OPP -
            CW CCW  - AND
          CENTER CW <> AND
                         HPP-GAS
 COLLISION IF CW ELSE OPP THEN
                    >PLNO
```

Здесь мы модифицировали приведенное выше правило SWAP-ON-DIAG, вставив оператор IF, который связан с оператором "COLLISION"[1].

Простота программного кода для правил SWAP-ON-DIAG и HPP-GAS является следствием инвариантности этих правил по отношению к повороту и того факта, что относительные соседи по

[1] В COLLISION мы проверяем, пуста или заполнена каждая диагональ; и если одна заполнена, то другая пуста. В случае HPP-GAS при столкновении мы поворачиваем блок против часовой стрелки, заменяя каждую клетку ее CW-соседом.

блоку CENTER, CW, OPP, CCW связаны с элементами блока независимым от поворота способом. Таким образом, четыре аналогичные (12.5) выражения могут быть сжаты в одно:

$$CENTER_{нов} - f(CENTER. CW, CCW, OPP)$$

следовательно, отпадает необходимость выбора одного из четырех выражений (см. п. 1) в зависимости от положения в блоке. Более того, такие "чувствующие позицию" слова для определения соседей автоматически выполняют работу п. 2.

Преимущество использования правила для блоков 2x2 состоит в том, что каждая плоскость битов предоставляет справочной таблице только четыре входных бита. Таким образом, чтобы просмотреть содержимое блока в *обеих* плоскостях 0 и 1, справочной таблице достаточно использовать 8 источников сигналов, а не 18, как в случае окрестности 3x3.

Познакомившись в разд. 9.1 и 10.1 с громоздкими методами, необходимыми для "программирования" динамики частиц в обычном клеточном автомате, читатель не может не оценить ясности описания и экономии средств, достижимых при использовании данного подхода. Вопрос не только в компактности *обозначений;* в CAM *реализация* существенно изоморфна концептуальной модели, и ее простота непосредственно переходит в эффективность моделирования.

12.5. Соседи по Марголусу

Как уже упоминалось, в CAM окрестность Марголуса поддержана специальными аппаратными средствами; это ведет к более эффективному использованию справочных таблиц и дает доступ к широкому кругу экспериментов.

Как было отмечено в разд. 11.1, две пространственные фазы &HORZ и &VERT сообща дают информацию, необходимую для разбиения массива клеток на решетку блоков 2x2. Фактически можно определить *блок* как группу из четырех смежных клеток, для которых значения объединенной пространственной фазы ш представлены так:

01
23

Два различных, связанных с четной и нечетной решеткой, способа объединения ячеек в блоки получают чередованием значений о и з, принимаемых &HV в начальной клетке массива (это осуществляется манипулированием <ORG-HV>, как показано в разд. 12.6).

В принципе можно было бы определить относительных соседей по блоку из предыдущего раздела в терминах обычных соседей клетки NORTH, N.EAST И Т. Д., обеспечиваемых окрестностью Мура. Например, CW, CCW, OPP функционально эквивалентны

```
                                        : CW
      &HV { EAST SOUTH NORTH WEST } ;
                                        : CCW
      &HV { SOUTH WEST EAST NORTH } ;
                                        : OPP
&HV { S.EAST S.WEST N.EAST N.WEST } ;
```

Другими словами, таблица для правила может быть запрограммирована так, чтобы объединить под управлением пространственных фаз восемь обычных соседей в три соседа по блоку (четвертым соседом по блоку является CENTER-клетка, которая используеся непосредственно). Однако чтобы таким способом синтезировать четыре соседа по блоку, пришлось бы использовать 11 из 12 входов таблицы (девять пространственных соседей и две пространственные фазы), что оставляет мало места для каких-либо дополнительных возможностей.

В CAM указанное выше объединение проводится непосредственно аппаратными средствами, так что CW, CCW, OPP, как и CENTER, соответствуют действительным сигналам на панели. Это отностся как к плоскости 0, так и к плоскости 1. Эти восемь сигналов (по 4 на каждую плоскость) связаны с восемью входами справочной таблицы посредством основного назначения *окрестности Марголуса* (см. табл. 7.2). Такое назначение дается в трех слегка отличных разновидностях, а именно:

N/MARG, которое порождает соседей

```
CENTER   CW    CCW   OPP
CEHNER'  CW'   CCW   OPP'
```

N/MARG-HV, порождающее соседей

```
CENTER   CW    CCW   OPP    HORZ
CENTER'  CW'   CCW'  OPP'   VERT
```

N/MARG-PH, которое порождает

```
CENTER   CW    CCW   OPP    PHASE
CENTER'  CW'   CCW'  OPP'   PHASE"
```

Назначение N/MARG оставляет на пользовательском коммутаторе два входа для введения пользователем дополнительных соседей (см. разд. 7.3.1). Две других разновидности окрестности дополняют десять входов основного назначения двумя псевдососедями. В частности, PHASE и PHASE' соответствуют двум добавочным фазовым сигналам, которые выдает компьютер-хозяин и которые аналогичны &PHASE и &PHASE' (предлагаемым дополнительным назначением &/PHASES), но они *отличаются* от них; они доступны только в окрестности N/MARG-PH. С другой стороны, псевдососеди HORZ и VERT, введенные основным назначением N/MARG-HV, ссылаются на *те же* сигналы, что и &HORZ и &VERT (которые предлагаются дополнительным назначением &/HV)[1]. Заметим, что в окрестности, выбранной назначением

N/MARG-PH &/PHASES,

пространственные фазы HORZ и VERT совсем не встречаются в качестве входов справочной таблицы; это не мешает CW и CCW соответствующим образом исполняться, поскольку этих соседей по блоку синтезируют аппаратурой, получая пространственно-фазовую информацию от сигналов HORZ и VERT независимо от их подключения к справочным таблицам.

Абсолютные соседи по блоку UL, UR, LL, LR возвращают значения четырех клеток блока согласно их абсолютной позиции внутри блока (см. разд. 12.4), при этом не имеет значения, где в блоке расположена сама клетка, запрашивающая эти значения. Непосредственно аппаратурой такие соседи не поддерживаются, однако они могут быть определены посредством относительных соседей по блоку CENTER, CW, CCW, OPP *и* двух пространственных фаз. Поскольку вся эта информация доступна справочным таблицам в окрестности N/MARG-HV, то когда выбрана эта окрестность, программное обеспечение делает доступными слова UL, UR и т. д. для использования при определении нового правила.

В заключение приводим слова соседства трех версий окрестности Марголуса:

N/MARG		
CENTER	CENTER'	CENTERS
CW	CW'	CWS
CCW	CCW'	CCWS
OPP	OPP'	OPPS

[1] **Поэтому не имеет смысла сопровождать основное назначение N/MARG-HV дополнительным назначением &/HV.**

N/MARG-PH		
CENTER	CENTER'	CENTERS
CW	CW'	CWS
CCW	CCW'	CCWS
OPP	OPP'	OPPS
PHASE	PHASE'	PHASES

N/MARG-HV					
CENTER	CENTER'	CENTERS	UL	UL'	ULS
CW	CW'	CWS	UR	UR'	URS
CCW	CCW'	CCWS	LL	LL'	LLS
OPP	OPP'	OPPS	LR	LR"	LRS
HORZ	VERT	HV			

12.6. Выбор четной/нечетной решетки

В разд. 11.3 мы обсудили, как построить двухфазное правило. Компьютер-хозяин переключал временную фазу с о на 1 и обратно в соответствии с выбранной схемой эксперимента, и это значение было доступно правилу в виде псевдососеда - &PHASE.

Похожая ситуация и в окрестности Марголуса. Однако значение временной фазы (скажем, о для шага, использующего четную решетку, и 1 для нечетной решетки) не сообщается непосредственно правилу; оно воздействует на правило, но только косвенным образом - путем "чувствительного к позиции" поведения соседей по блоку, которое знает, какая решетка используется в данный момент.

Как было отмечено ранее, OPP, CW и другие соседи по блоку "знают" свое относительное положение в блоке благодаря внутренним пространственно-фазовым сигналам HORZ и VERT. Значения сигнала HORZ о и 1 чередуются *в пространстве* при горизонтальном движении по массиву, а сигнала VERT - при вертикальном. Объединенное значение, которое эта пара сигналов принимает в начале координат массива, определяет, какая из двух решеток выбрана на данном шаге. Значение <о,о> соответствует четной решетке, <1,1> - нечетной. Таким образом, для того, чтобы переключиться с одной решетки на другую, необходимо изменять *во времени* (т. е. от шага к шагу) с о на 1 и обратно значения <ORG-H> и <ORG-V>, как объясняется в разд. 11.2. Это осуществляется путем исполнения следующего рабочего цикла:

```
                : ALT-GRID
    0 IS <ORG-HV> STEP
    3 IS <ORG-HV> STEP
```

12.7. Фазочувствительиый газ

В предыдущих примерах один и тот же рецепт применялся как для четных, так и для нечетных шагов; однако если временная фаза, которую используют для выбора одной из двух решеток, явно доступна в качестве псевдососеда, то появляется возможность чувствовать эту фазу изнутри правила и пользоваться разными рецептами для разных значений фазы. Таким образом, динамика клеточного автомата, использующего окрестность Марголуса, может "модулироваться" временной фазой не только косвенно - через чередование четной и нечетной решеток, - но и явно через само правило. Соответствующий рабочий цикл для этого будет следующим:

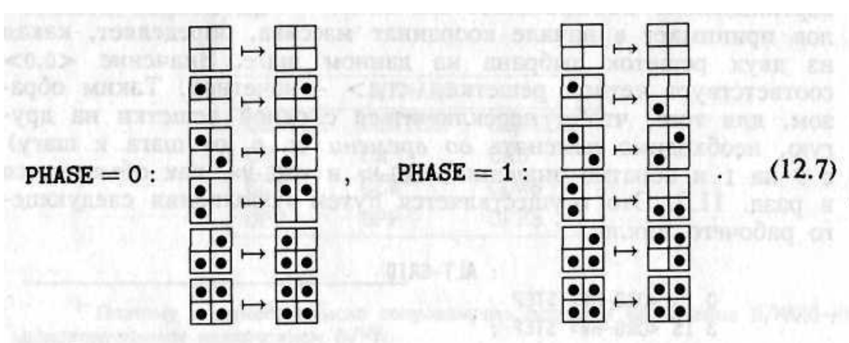
```
                                   ALT-GRID-PH
0 IS <0R6-HV> 0 IS <PHASE> STEP
3 IS <ORG-HV> 1 IS <PHASE> STEP
```

Здесь мы опишем еще один механизм переноса частиц, в котором частицы движутся горизонтально и вертикально, а не по диагонали (как в разд. 12.2). Иногда удобно иметь в наличии оба типа механизма переноса (см. разд. 15.6).

Мы будем пользоваться окрестностью N/MARG-PH и приведенным выше рабочим циклом ALT-GRID-PH. Сначала, по аналогии с разд. 12.2, мы построим механизм равномерного движения частиц без столкновений. Движение будет осуществляться вдоль строк и столбцов массива. Правило для блока состоит в следующем:

> ROT-CW/CCW. На четном шаге (PHASE=0) повернуть содержимое всего блока *по часовой стрелке;* на нечетном шаге (PHASE=1) повернуть *против часовой стрелки.*

Другими словами, в зависимости от значения PHASE использовать одну из двух таблиц:



На языке FORTH это правило выглядит так:

```
                    : ROT-CW/CCW
PHASE { CCW CW } > PLNO ;
```

Фактически, чтобы повернуть весь блок по часовой стрелке, достаточно попросить каждую клетку сделать копию своего соседа CCW (см. разд. 5.1), а чтобы повернуть блок против часовой стрелки - копию своего соседа CW. Направление поворота определяется значением PHASE.

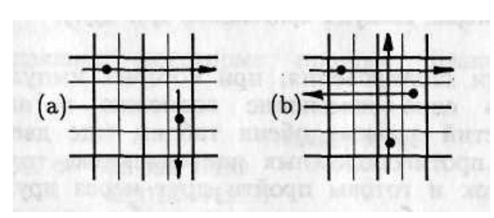


Рис. 12.4. ROT-CW/CCW. (а) Однородное движение частицы. (Ь) Частицы проходят одна через другую без столкновения.

Давайте изучим это правило более детально.

Возьмем изолированную частицу. Для первого шага выберем, скажем, четную решетку, и рассмотрим клетку, лежащую в левом верхнем углу блока (рис. 12.4а). Путем поворота на четверть круга по часовой стрелке правило заставит частицу двигаться в правый верхний угол.

На следующем шаге используем нечетную решетку. Здесь частица обнаруживает себя в левом нижнем углу и вращение блока против часовой стрелки передвинет ее в правый нижний угол.

Мы завершили цикл из двух шагов, и частица переместилась на две позиции вправо. К следующему шагу мы оказываемся в той ситуации, с которой все начиналось.

ч

Чередуя решетки, идем далее. Каждая частица будет равномерно двигаться по горизонтали или вертикали. В каком из четырех направлений она будет двигаться, определяется начальным положением частицы.

Как и в случае правила SWAP-ON-DIAG, частицы, которые встречаются лоб в лоб или под углом 90°, не замечают друг друга и продолжают двигаться в прежних направлениях. Правило перемещает все, что ни попадется - вакуум или частицы - ему все равно.

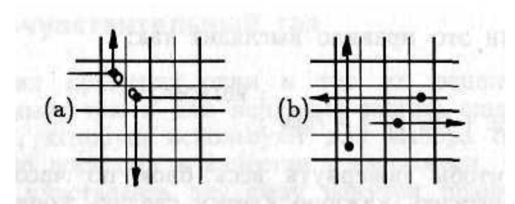


Рис. 12.5. TM-GAS. (а) Частицы, сталкивающиеся внутри блока, отскакивают под углом 90 градусов; маленький кружок - напоминание о том, что частицы остаются на одном месте в течение одного шага. (Ь) Столкновения невозможны для частиц, летящих мимо друг друга в соседних блоках, или для частиц, которые движутся ортогонально друг другу.

Чтобы ввести столкновения, при которых импульс сохраняется, опять лишь одно изменение возможно в наших таблицах (12.7) - третий элемент обеих таблиц (где две частицы, движущиеся в противоположных направлениях, только что вошли в общий блок и готовы пройти друг через друга) может быть изменен так, чтобы, вместо того чтобы продолжать двигаться в прежних, частицы выйдут из столкновения в противоположных направлениях вдоль *другой* оси массива (т. е. если двигались по горизонтали до столкновения, то по вертикали - после и наоборот). Новый элемент таблиц есть

(12.8)

т. е. при столкновении двух частиц соответствующий блок не поворачивают. На языке Forth CAM это выглядит следующим образом:

```
                    : TM-GAS
            COLLISION IF
             CENTER ELSE
PHASE { CCW CW } THEN >PLNO ;
```

Здесь мы изменили правило ROT-CW/CCW, вставив оператор IF, который связан с входной меткой "столкновения". Условия для COLLISION такие же, как для HPP-GAS (разд. 12.3), но результат столкновения другой. Работа этого правила показана на рис. 12.5.

Заметим, что в TM-GAS частицы, которые движутся относительно друг друга по соседним строкам или столбцам, не обязательно сталкиваются: если в данный момент выбрана не та решетка, то частицы попадают в разные блоки и вообще не замечают друг друга.

12.8. Примеры

В этом разделе мы дадим несколько примеров интересных правил, которые могут быть получены при использовании схемы окрестности по Марголусу. Основные примеры (включая моделирование с помощью клеточного автомата вычислений, для которых этот метод первоначально и был разработан) даны в части III, посвященной физическому моделированию.

12.8.1. Фракталы

Правило, порождающее некоторые простые фракталы (см. [34]), имеет вид

```
                                      : FORGET-ME-NOT
              CW CCW OPP XOR XOR >PLNO
                         CENTER >PLN1 ;
```

(здесь и в дальнейших правилах для окрестности Марголуса мы подразумеваем использование таких рабочих циклов, которые обеспечивают требуемое чередование решеток). Фотография 10 иллюстрирует работу этого правила начиная с небольшого квадрата единиц в центре экрана (ECHO в плоскости битов 1 служит для добавления некоторого цвета).

Любопытно, что существует правило второго порядка, которое проявляет аналогичное поведение, хотя и использует *окрестность Мура:*

```
N/MOORE
                                       : ME-NEITHER
      N.EAST  N.WEST  S.EAST  S.WEST
                     XOR XOR XOR
                     CENTER' XOR >PLNO
                         CENTER >PLN1 ;
```

На самом деле существуют некие формальные связи между обратимыми правилами, полученными посредством приемов ок- >ирестности Марголуса, и правилами, полученными при использовании методов второго порядка (см. разделы 14.2, 14.3 и 17.7).

Если вы повернете последнее *правило* на 45 градусов, записав его в виде

```
                                       : NOR-HE
             NORTH SOUTH WEST EAST
                     XOR XOR XOR
                     CENTER' XOR >PLNO
                         CENTER >PLN1 ;
```

то вновь получите в сущности то же поведение при условии, что начальная *конфигурация* также повернута на 45 градусов (т. е. начинаем с ромба, а не с квадрата).

12.8.2. Криттеры

Правило FORGET-ME-NOT из предыдущего раздела обратимо и может быть отработано в обратном направлении очень просто: нужно использовать одну и ту же решетку два раза подряд, а затем возобновить чередование решеток (как более детально разъяснено в разд. 14.5). Этот прием срабатывает, потому что в этом правиле трансформация, приложенная к блоку, совпадает с обратной к ней. Большинство обратимых правил для блока, с которыми мы будем иметь дело в этой книге, включая и те, которые используют правило HPP-GAS разд. 12.3, обладают данным свойством. Здесь же, однако, нам хотелось бы представить весьма интересное обратимое правило, которое этим свойством не обладает. Вот оно:

```
                        : -CENTER
           CENTER I XOR ;
                        : -OPP
              OPP 1 XOR ;
                        : CRITTERS
   CENTER OPP CW CCW + + +
 { -CENTER -CENTER CENTER
         -OPP -CENTER } >PLNO ;
```

То есть берется дополнение этого блока, если только он не содержит ровно две единицы: если же он содержит две, то его оставляют без изменений. Блок, содержащий три единицы, кроме взятия дополнения еще и поворачивают на 180 градусов. Обратное правило таково:

```
                        : CRITTERS*
    CENTER OPP CW CCW + + +
    { -CENTER -OPP CENTER
          -CENTER -CENTER } >PLNO ;
```

Поскольку CRITTERS поворачивает и дополняет блок с *тремя* единицами, то обратное правило должно дополнять блок с *одной* единицей и поворачивать его на -180 градусов.

Поскольку преобразование блока, выполняемое CRITTERS, не совпадает с обратным к нему преобразованием, то, чтобы заставить систему развиваться в обратном направлении во времени, недостаточно пройти в обратном направлении последовательность чередований решетки. Необходимо еще использовать обратное преобразоваеие - CRITTERS*. Однако мы можем вое-

пользоваться самим CRITTERS и все же увидеть систему проделывающей обратный путь, если надеть "дополняющие очки". Фактически это правило не меняется при обращении времени (см. разд. 14.1), сопровождаемом операцией *дополнения:* каждое состояние "обратного" движения, полученного дополнением конечного состояния и применением правила CRITTERS, есть просто дополнение соответствующего состояния действительного обратного движения, получаемого при уходе из конечного состояния (без его дополнения) с применением CRITTERS*.

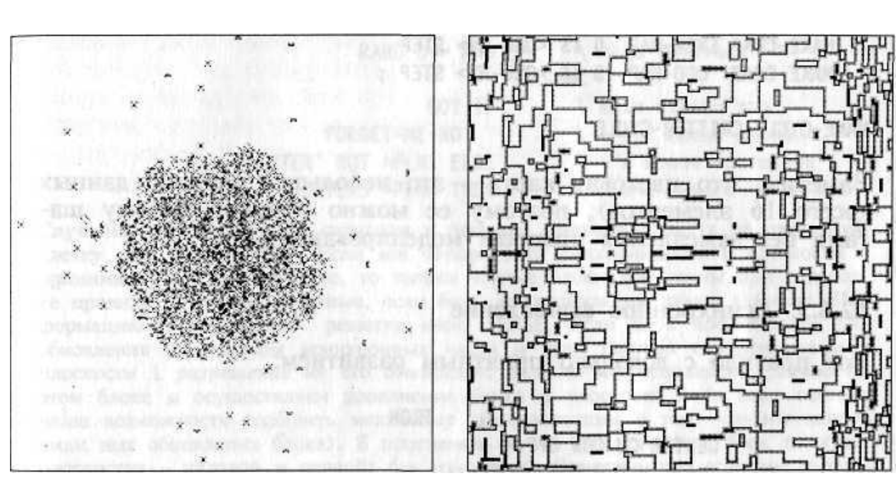


Рис. 12.6. (а) Криттеры расползаются из аморфного куска вещества; (Ь) Хореография правила TRON.

CRITTERS напоминает LIFE (рис. 12.6 и фото 11) разнообразием порождаемых структур и их живостью. В силу своей обратимости он не может создать порядок из хаоса (это, попросту говоря, и есть второй закон термодинамики); однако если мы начнем с весьма неоднородного начального состояния (например, островок беспорядка на фоне нулей), то мы увидим богатую картину развития. Небольшие "криттеры" быстро движутся горизонтально и вертикально. При столкновении они мЪгут отскочить назад или повернуть под углом 90°; иногда они склеиваются - до следующего столкновения, когда они вновь разделятся - или даже скапливаются вместе, чем-то напоминая небольшие части электронных схем.

Поскольку области из одних нулей или единиц дополняются на каждом шаге, то на экране отображается непрерывное мерцание, что весьма раздражает. Для устранения этого мерцания вы можете воспользоваться рабочим циклом, в котором

на четном и нечетном шагах применяются различные цветовые карты:

```
                                          : EVEN-MAP
          0 >REO             0 >6REEN
      ALFA >BLUE         ALPHA >INTEN  ;
                                          : ODD-MAP
          0 >RED             0 >GREEN
   ALPHA NOT >BLUE   ALPHA NOT >INTEN  ;
                                          : CRITTER-CYCLE
MAKE-CMAP  EVEN-MAP   0  IS  <0R6-HV>  STEP
MAKE-CMAP   ODD-MAP   3  IS  <ORG-HV>  STEP  ;

MAKE-CYCLE  CRITTER-CYCLE
```

Заметим, что цветовая карта - это небольшая таблица данных (всего 16 элементов), поэтому ее можно изменять между шагами без замедления процесса моделирования.

12.8.3. Асинхронное вычисление

Вот правило с довольно приятным развитием:

```
                               : TRON
       CENTER CW CCW OPP + + +
         { 1 U U U 0 } >PLNO   ;
```

(где и, как и прежде, означает сокращение для CENTER). Это правило дополняет блок, если только все элементы блока одинаковы, в противном случае оно оставляет содержимое блока прежним. Богатая хореография линейных конфигураций будет возникать на фоне нулей из простых конфигураций с прямолинейными краями (рис. 12.6Ь).

Оказывается, что это правило также обеспечивает паттерны битов, которые по аналогии с разд. 9.6 могут быть использованы как "фазы" при управлении эволюцией асинхронного клеточного автомата, но с упрощенной схемой, допускаемой окрестностью Марголуса (см. также [37]). Вкратце, предположим, что мы запускаем *любое* правило, которое использует окрестность Марголуса (например, FORGET-ME-NOT), в плоскости О, и правило TRON в плоскости 1 (разумеется, заменив CENTER на CENTER' и т. д.). Как будет объяснено в разд. 15.7, САМ-В предоставит подходящие для окрестности Марголуса часы Пуассона, т. е. всем четырем клеткам блока будет доступен *один и тот же* случайный бит.

```
N/MARG-PH «./CENTERS
                                          : ASYNC-CYCLE
        0 IS <ORG-HV> 0 IS <PHASE> STEP        \ чередование и
        3 IS <ORG-HV> 1 IS <PHASE> STEP  ;     \  решетки и фазы
                                          : RAND
                                &CENTER'  ;    \ асинхр. часы
                                          : GO?
          PHASE CENTER' = PHASE  CW  =
          PHASE     CW = PHASE  OPP" =
                             AND AND AND       \ могу обновить?
                            RAND 0<> AND  ;    \ надо пытаться?
                                          : ASYNC
                                  GO? IF       \ если сделаю это
                         FORGET-ME-NOT         \   тут новые данные
                  CENTER' NOT >PLN1 ELSE       \   и новое состояние
                      CENTERS >PLNA THEN  ;    \ иначе ничего не менять
```

Случайный стимул станет сигналом к попытке обновить блок (а не отдельную клетку, как в разд. 9.6). Если все четыре бита синхронизации в плоскости 1 принимают нулевое значение, то только четные блоки допустимы при отработке правила, и только нечетные, если биты синхронизации равны единице. Информацию о выбранной решетке несет PHASE. Если блок *подготовлен* для обновления посредством асинхронных часов и если паттерн синхронизации в плоскости 1 *разрешение* на его обновление дал, то мы исполняем правило на этом блоке *и* осуществляем дополнение блока в плоскости 1 (в этом состоит наша возможность сообщить механизму синхронизации о том, что мы закончили шаг обновления блока). В противном случае мы оставляем блок (в обеих плоскостях - нулевой и первой) без изменений. Заметим, что если мы выполняем это правило детерминированно (т. е. если RAND всегда возвращает 1), то эволюция фазовых битов в плоскости 1 - та, что предписана правилом TRON.

Давайте начнем с заполненной нулями плоскости 1: для простоты считаем, что в начале все узлы асинхронной сети находятся в одной фазе. Вычисление в плоскости 0 будет проводиться с холмами и низинами времени - местами, в которых обновление проводится чаще, чем в соседних, или реже; причем крутизна склонов ограничена и на них нет обрывов. Если паттерн в плоскости 1 не сохраняет информации об очередности (кто кого опережает), то эта деятельность не может быть распознана как вычисление, которое мы имели в виду (а именно, FORGET-ME-NOT). Другими словами, паттерн битов в плоскости 1 - это контурная карта холмов и низин времени.

Если в некоторый момент мы заменим определение RAND, положив его значение всегда равным 1 (например, заменяя это правило в CAM-B на тождественное правило и заполняя плоскость 2 единицами), то холмы и низины выровняются, и вы обнаружите, что синхронизация действительно корректна: мы получим паттерн в CAM-A, в котором все клетки обновляются в точности так, как если бы правило FORGET-ME-NOT исполнялось все время синхронно.

12.8.4. Цифровая логика

Здесь мы представим простое правило, которое позволяет непосредственно моделировать обычную цифровую логику. Ясно, что любое правило, которое может моделировать элементы цифровой логики общего вида, может моделировать и цифровой компьютер. Доказать, что такое правило, как LIFE, может моделировать цифровой компьютер - это что-то вроде *tour de force* - нечто подобное показу возможности создания компьтера на основе столкновений биллиардных шаров! Более скромный подход - создание правила, которое точно может исполнять логику; тогда и доказать это будет проще.

Наше правило будет работать следующим образом. В плоскости 1 нарисуем линии плотностью в одну клетку, которые идут от места к месту; это - наши провода. В плоскости О поместим частицы, которые движутся вдоль проводов, и в некоторых узлах взаимодействуют друг с другом. Это - наши сигналы.

```
NEW-EXPERIMEHT   N/MARG
                                     ALT-GRID
                EVEN-GRID STEP          \ правило чередования решеток
                 ODD-GRID STEP
                                     #WIRES ( - 0..4)
    CENTER' CW CCW OPP" + + + ;         \ число проводов блока
                                     SIGNALS
               OPP' IF OPP THEN         \ сигналы идут по
               CW   IF CW  THEN         \  проводам
               CCW' IF CCW THEN ;
                                     CONTROLS
            OPP'  0= IF OPP THEN        \ AND? OR? для 3WIRE
            CW'   0= IF CW  THEN
            CCW'  0= IF CCW THEN ;
                                     1WIRE ( -- 0|1)
               OPP CW CCW OR OR ;
                                     2WIRE ( - 0|1)
       SIGNALS CW CCW OR IF             \ проверка для NOT (SIGNALS
                   OPP XOR THEN ;       \  дает здесь один выход
                                     3WIRE
                   CONTROLS IF
              SIGNALS AND ELSE          \ SIGNALS выдает 2 значения
              SIGNALS  OR THEN ;        \  при трех проводах
                                     LOGIC
                    CENTER* IF
                      WIRES
  { 0 1WIRE 2WIRE 3WIRE OPP }
                           ELSE
```

```
                CENTER THEN  >PLNO
                   CENTER'   >PLN1 ;

MAKE-TABLE LOGIC
MAKE-CYCLE ALT-GRID
```

Согласно этому правилу, сигналы распространяются по проводам; по какому пути они распространяются - зависит от начального выбора решетки. В блоке, содержащем две части провода с сигналом на одной из них, этот сигнал переходит на другую часть провода (рис. 12.7a). Следовательно, сигналы могут распространяться горизонтально, вертикально или по диагонали - только вдоль проводов.

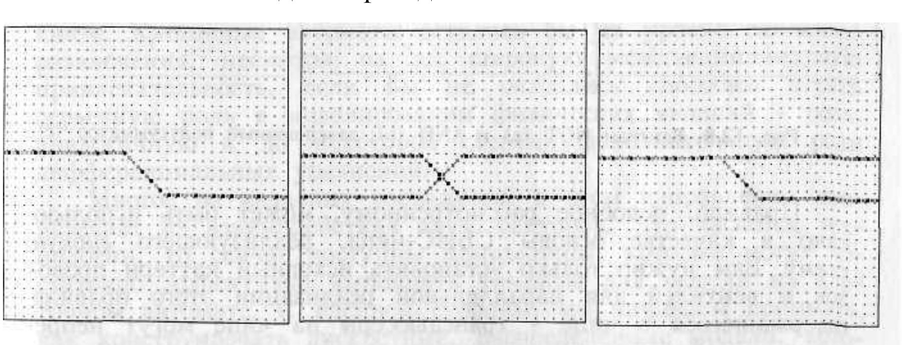


Рис. 12.7. Распространение, пересечение и ветвление сигнала.

Если вам нужно выполнить пересечение сигналов (рис.12.7Ь), то просто осуществите пересечение соответствующих проводов: не заметив друг друга, сигналы разойдутся (обеспечьте пересечение проводов в центре блока с тем, чтобы в блоке было четыре части провода).

Если вам нужно ветвление (рис. 12.7с), то просто расщепите провод: копия сигнала будет проходить по обеим частям (расщепляйте провод в центре блока с тем, чтобы в блоке было три части провода).

При выполнении логической операции сигнал, находящийся *рядом* с прямолинейным участком провода (но не *на* нем), будет заморожен в этом месте и будет выполнять операцию дополнения сигналов, проходящих по данному участку (рис, 12.8а). Такой сигнал, расположенный у развилки на проводе (рис. 12.8Ь), превращает вилку в AND-вентиль - каждый из трех концов всегда будет логическим "и" (AND) двух других входов.

Наконец, для иллюстрации компактности схемы, которая достигается при использовании данного правила, мы приводим на рис. 12.8с схему двоичного полусумматора.

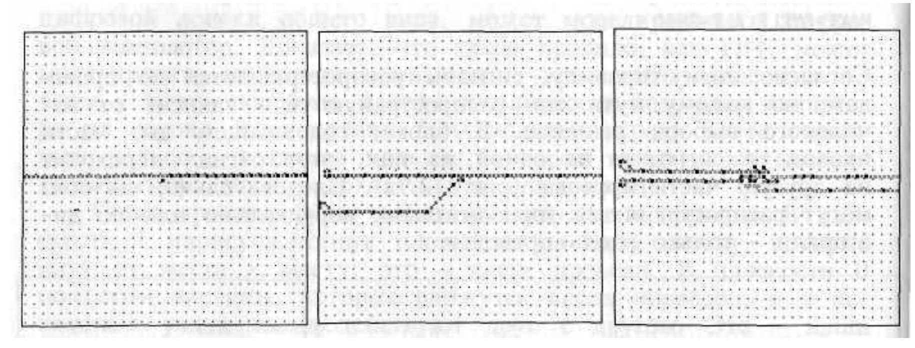


Рис. 12.8. Вентили NOT (a) **и** AND (b); завершенный полусумматор (c).

Правило, подобное рассмотренному, может быть использовано в качестве основы СБИС-чипа, реализующего *гибкую схему*. Вам нужно только установить исходный паттерн проводов и вентилей для параллельной реализации этого правила, содержащегося в чипе - трансдьюсеры на чипе могут непрерывно преобразовывать несколько входов и выходов, превращая их клеточные представления в чипе в уровни напряжения на выводах, и обратно. Асинхронная реализация, использующая метод предыдущего раздела, может работать весьма быстро.[1]

[1] Брайан Силверман и Уоррен Робиннетт независимо изобрели правила клеточного автомата, которые могут быть подходящими для таких гибких (программируемых) схем. Правила Роджера Бэнкса, одно из которых рассматривалось в разд. 5.5, также могут быть пригодны для этой цели.

Часть III. Физическое моделирование

Глава 13

СИМПТОМЫ И ПРИЧИНЫ

Entia non stint multiplicanda praeter necessltatem[1]

["Бритва" Оккама]

Часть, посвященная *физическому моделированию,* которую мы сейчас начинаем, является концептуально важнейшей в этой книге. Клеточные автоматы остались бы на уровне салонного развлечения, несмотря на их широкую междисциплинарную привлекательность, если бы не оказались способны играть серьезную роль в моделировании физических явлений - роль, аналогичную и до некоторой степени дополнительную роли дифференциальных уравнений.

Многое из недавних достижений в этом направлении проистекает из взаимодействия двух факторов. Что касается технологии, то сейчас доступны компьютеры, которые могут достаточно эффективно выполнять команды для модели на клеточном автомате. С идейной стороны, мы начинаем понимать, как конструировать дискретные, распределенные модели, которые способны отразить существенные черты физической причинности, такие, как микроскопическая обратимость.

13.1. Мелкозернистые модели физических явлений

Наука занимается объяснением явлений. Многие из систем, которые нас интересуют, не сопровождаются описательными руководствами и чертежами, и их нельзя рассмотреть изнутри или разобрать на части без нарушения их поведения. Мы говорим, что "понимаем" сложную систему, когда можем построить из простых компонент, которые нам уже хорошо понятны, модель, которая ведет себя аналогично.

Бели имеющийся в нашем распоряжении набор компонент слишком богат, то зачастую очень легко получить модели, демонстрирующие ожидаемое поведение, просто потому что сами внешние признаки, а не какие-то более глубокие внутренние причины, были непосредственно запрограммированны подобно специальным эффектам в компьютерных играх.

[1] **Не умножайте сущности без надобности** ***(лат.)*.**

В науке мало пользы от моделей, которые рабски подчиняются всем нашим желаниям. Мы хотим иметь модели, которые дерзят нам; модели, которые имеют свой собственный ум. Мы хотим получить из наших моделей больше, чем мы в них вложили. Разумная отправная точка - вложить *как можно меньше*.

Чем проще примитивы, используемые для описания сложной системы, тем больше бремя вычислений, необходимых для достижения точного и детального предсказания на основе модели[1]. По этой причине развитие математики в определенный период времени отражает природу вычислительных средств, доступных в этот момент, в значительно большей степени, чем можно было бы предположить. В прошедшие три столетия огромное внимание было уделено моделям:

(1) определенным на *континууме* и хорошо ведущим себя на нем,

(2) *линейным,* и

(3) включающим небольшое число *агрегированных* переменных.

Это внимание отражает скорее не предпочтения самой природы, а тот факт, что человеческий мозг, вооруженный только карандашом и бумагой, действует лучше всего, когда манипулирует небольшим количеством символических объектов, обладающих значительной концептуальной глубиной (например, вещественные числа, дифференциальные операторы). В связи с этим имеется тенденция к сосредоточению усилий на проблемах, подходящих для того, чтобы дать символическое решение в законченном виде.

С появлением цифровых компьютеров область, в которой удобнее всего работать, сместилась. Хотя значительный прогресс все еще может быть достигнут в указанных выше более традиционных областях, горизонт резко расширился в других направлениях, а именно:

(Г) *дискретные* модели,

(2') *нелинейные* модели, и

(3') модели, включающие большое число *распределенных* переменных.

[1] Например, химия сводима в принципе к квантовой механике, но на этой основе были вычислены во всех деталях только структуры простейших атомов. Для более сложных веществ необходимо прибегнуть к помощи аппроксимаций, эмпирических методов или феноменологических теорий более высокого уровня.

В таких моделях большее внимание уделяется обработке большого числа объектов простой природы (например, булевых переменных и логических функций) - задача, в которой компьютеры особенно эффективны.

Клеточные автоматы пытаются извлечь выгоду из логического предела этого подхода. Они сокращают количество примитивных составляющих, которые могут входить в модель, до одной, а именно "единичной клетки", управляемой простым правилом и соединенной с идентичными ей клетками однородным шаблоном взаимосвязей. Задача состоит в том, чтобы подобрать это правило и этот шаблон так, чтобы все, что мы хотим от модели в широком диапазоне масштабов, следовало бы из одного этого выбора.

Будет ли этот подход работать, зависит, конечно, от того, что мы пытаемся моделировать. Физики столетиями использовали рабочую гипотезу, что "мир в основе своей прост - только его очень много". Если это справедливо, то клеточные автоматы и машины клеточных автоматов могут представить полезный инструмент в попытках понять и описать природу.

Глава 14

ОБРАТИМОСТЬ

Так вот, если мы представим нашу Вселенную несколько миллионов лет назад, то получим объект, похожий на Дональда Дака. Очевидно, здесь где-то ошибка.

[С. Харрисон]

Насколько известно, обратимость свойственна всем физическим законам. Она, в частности, является необходимым условием выполнения второго закона термодинамики[1] и достаточным условием для существования сохраняющихся величин[2].

Некоторые основные черты физических законов, такие как *однородность* и *локальность,* присущи клеточным автоматам по построению; с другой стороны, *обратимость* не получается автоматически и должна быть представлена программно. Еще совсем недавно никто не знал, как это делать систематически. Было даже подозрение, что обратимость можно ввести лишь за счет утраты других свойств (таких как вычислительная и структурная универсальность), существенных для общецелевого моделирования.

Оказывается, что можно конструировать нетривиальные клеточные автоматы, которые проявляют *полную* обратимость (в этом отношении они свободны от погрешностей, вызванных аппроксимациями, столь частыми в традиционном численном моделировании). Тем самым можно получить модели, хотя и очень далекие от реальности в других отношениях, но полностью сохраняющие эту фундаментальную особенность физического процесса.

[1] Для локально взаимодействующих систем, таких как клеточные автоматы, имеющих конечное количество информации на одно место, обратимость *эквивалентна* второму закону термодинамики.

[2] В физике обратимая система, имеющая *n* степеней свободы, обладает 2n-1 сохраняющимися величинами, часть из которых (например, энергия, импульс и т. д.) имеет особое значение ввиду их связи с фундаментальными симметриями физических законов [33]. Соображения, которые приводят к этим законам сохранения, могут быть обобщены на клеточные автоматы. Основная идея состоит в том, что произвольное данное состояние "кодирует" всю информацию, необходимую для определения частной динамической траектории, на которой оно лежит. Если система обратима, то эта информация не теряется в процессе эволюции.

14.1. Обратимые клеточные автоматы

Клеточный автомат является *детерминированной* системой, т. е. для каждого возможного состояния всего клеточного автомата правило определяет одно и только одно последующее состояние. Можем ли мы запустить клеточный автомат "в обратном направлении", т.е. по заданному правилу сконструировать новое, которое заставит систему проделать ее шаги в обратном порядке? Ясно, что в общем случае это возможно, только если система, определенная исходным правилом, к тому же *детерминирована в обратном направлении,* т. е. если для каждой возможной конфигурации клеточного автомата существует одна и только одна предшествующая.

Система, детерминированная в обоих направлениях времени, называется *обратимой,* а правило, которое заставляет ее двигаться назад во времени, называется *обратным* по отношению к исходному, или *прямому* правилу. (Термин "инвертируемый", предпочитаемый математиками, эквивалентен термину "микроскопически обратимый", или просто "обратимый", используемому физиками в этом смысле.)

Заметим, что за исключением тривиальных случаев, обратное правило, когда оно существует, *отлично* от прямого. Например, после наблюдения в течение некоторого времени за выталкиванием вверх изображения клеточного автомата правилом SHIFT-NORTH из разд. 5.1, мы можем просмотреть весь процесс в обратном направлении, используя обратное правило

```
                : SHIFT-SOUTH
      NORTH >PLNO ;
```

несомненно отличное от SHIFT-NORTH.

В специальных случаях, тем не менее, возможно достичь практически такого же эффекта "обратимости во времени" за счет выполнения *того же* прямого правила, что и при прямом направлении времени, но используя в качестве начального состояния для обратного движения состояние, полученное из конечного состояния при продвижении вперед предписанным преобразованием - процедурой "транслитерации".

Динамический закон, обладающий этим свойством, называется *инвариантным к обращению времени относительно заданного преобразования состояний,* или просто *инвариантным к обращению времени,* когда оператор преобразования известен.[1] Таким образом, "инвариантность к обращению времени" более сильное свойство, чем просто "обратимость".

[1] В ньютоновской механике, например, процедура обращения времени состоит из обращения импульсов всех частиц.

Теория обратимых клеточных автоматов содержит много открытых проблем [58]. Неизвестна, в частности, общая процедура определения того, имеет ли данное правило обратное (возможно, она неразрешима). Тем не менее для двух заданных наперед правил всегда можно в принципе решить, являются ли они обращениями друг друга. В соответствии с представленным ниже методом конструирования обратимых клеточных автоматов, прямое правило всегда сопровождается обратным ему, так что никогда нет никаких сомнений, относящихся к его обратимости - ни в принципе, ни на практике.

14.2. Метод второго порядка

Первый общий метод, который мы рассмотрим для получения обратимых клеточных автоматов, включает создание систем второго порядка, являющихся инвариантными к обращению времени. Пример этого метода был дан в разд. 6.2, естественным продолжением которого является настоящее обсуждение.

Перефразируя Зенона, если мы вырежем один кадр из фильма, изображающего летящую пулю, то не сможем по нему узнать, что пуля делает. Если же даются два последовательных кадра этого фильма, то можно вычислить траекторию пули. Таким образом, из этих двух кадров, интерпретируемых как "прошлая" и "настоящая" позиции пули, мы можем сконструировать третий, задающий ее будущую позицию. Эту процедуру можно повторить. Законы ньютоновской механики таковы, что если из каких-то соображений поменять эти два кадра местами, то мы придем к вычислению траектории *в обратном направлении*.

Следующий общий метод конструирования клеточных автоматов, который работает аналогично, предложил Эд Фредкин из Массачусетского технологического института.

Начнем с динамической системы, в которой последовательность конфигураций, через которые она проходит, задается итерацией зависимости вида

$$c^{г+1}=mc\backslash \qquad (14.1)$$

где конфигурации c можно пока что представлять себе, например, как вещественные числа, t является *динамическим законом* системы, т. е. функцией, которая в качестве аргумента берет текущую конфигурацию с' и возвращает следующую конфигурацию $c^{м}$. В общем случае закон (14.1) приводит к необратимой динамике.

А теперь рассмотрим новую систему, определенную зависимостью

$$c^{t+1} = \tau c^{t} - c^{t-1}. \qquad (14.2)$$

Это пример системы *второго порядка,* в которой "следующая" конфигурация c^{*+1} является функцией и "текущей" конфигурации, и "прошлой" конфигурации $c^{*''1}$ (таким образом, она использует *пару* последовательных конфигураций, чтобы полностью определить дальнейшую траекторию). В общем случае зависимости второго порядка приводят также и к необратимой динамике. Тем не менее, зависимость второго порядка специального вида (14.2) гарантирует обратимость динамики для *произвольного.* Действительно, разрешая (14.2) относительно $c^{'''1}$, получаем зависимость

$$c^{t-1} = \tau c^{*} - c^{*+1}; \qquad (14.3)$$

т. е. пары последовательных конфигураций достаточно для однозначного определения также и *обратной* траектории.

Заметим, что в этом случае одной конфигурации недостаточно, чтобы иметь возможность продолжить траекторию. Полное описание динамического состояния системы теперь представляется *упорядоченной парой* конфигураций в виде (а, *Ь)* (в (14.1) одна конфигурация сама по себе являлась *состоянием* системы)[1].

Тот факт, что для системы, определенной уравнением (14.2), обратная траектория существует и единственна, означает, что система *обратима;* то, что ее поведение при движении в обратном направлении может быть вычислено применением прямой динамики к подходящим образом определенным состояниям, обращенным во времени, означает, что система инвариантна по отношению к такой операции обращения времени. Обращение времени определяется здесь как операция, которая меняет местами две конфигурации, образующие состояние, т. е. $(c0,c_a)$ заменяется операцией обращения времени на$(c_a,c/з)$.

Приведенные выше рассуждения можно непосредственно распространить на клеточные автоматы. Пусть в уравнении

[1] Возможна система, *не* являющаяся системой второго порядка, состояние которой - пара конфигураций. Тот факт, что динамика имеет *второй порядок,* выражается следующим ограничением: состояние *<c,d>* может следовать за состоянием *<a,Ь>,* только если *Ь=с* .

(14.1) различные с суть конфигурации клеточного автомата, а г- произвольное правило клеточного автомата. Если наши клетки имеют га состояний {0, 1,..., га-1}, то операция "-" в (14.2) может быть взята по mod га, а правило второго порядка становится следующим:

- Для каждой клетки применить исходное правило к окрестности "настоящей" клетки.
- Прежде чем принять этот результат в качестве "будущего" состояния клетки, вычесть ее "прошлое" состояние[1].

Обратимые правила второго порядка могут быть сконструированы с использованием в уравнении, подобном (14.2), других операций, чем вычитание. Вы можете даже сделать решение о том, какую операцию использовать, зависящим от соседей в момент *L* В обратимом правиле второго порядка наиболее общего вида окрестность в момент *t* используется для того, чтобы выбрать перестановку на множестве состояний {0,1,...,п-1}. Клетка применяет эту перестановку к своему предыдущему состоянию, чтобы сконструировать ее следующее состояние [62].

В примерах гл. 6 два последовательных кадра, которые вместе определяют состояние системы, запоминаются в плоскостях битов 0 ("настоящее") и 1 ("прошлое"). На каждом шаге мы создаем "будущую" конфигурацию из содержимого плоскостей 0 и 1 и помещаем ее в плоскость 0, в то время как текущее содержимое плоскости 0 перемещается в плоскость 1. Таким образом, плоскости 0 и 1 всегда содержат два последних кадра.

В CAM можно легко обращаться с клетками, имеющими четыре состояния, запоминая существующее состояние в CAM-A (плоскости 0 и 1), а прошлое состояние в CAM-в (плоскости 2 и 3). Соединение настоящего и прошлого осуществляется назначениями окрестностей

```
CAM-A &/ CENTERS
CAM-B &/ CENTERS ,
```

где CAM-A может просматривать прошлое в CAM-в и использовать его для вычисления будущего, в то время как CAM-B может просматривать настоящее в CAM-A И сохранять его как прошлое для следующего шага.

[1] В примерах разд. 6.2-6.3 клетки имели только два состояния, и поэтому "*n* mod 2" могло быть записано как XOR.

14.3. Чередующиеся подрешетки

Обозначим *c\j* состояние в момент времени *t* клетки, имеющей координаты г, *j*. Особая ситуация возникает, когда только четыре соседа *север, юг, запад* и *восток* входят в качест-ве аргументов функции *m* , так что уравнение (14.2) принимает вид

Эта зависимость соединяет только те места пространства-времени, для которых число $i+j+t$ имеет одинаковую *четность,* т. е. места, принадлежащие одной и той же подрешетке (черной или белой) шахматной доски пространства-времени. Таким образом, система состоит из двух *независимых* подсистем (одна из которых работает на черной подрешетке, а одна на белой), имеющих идентичные динамические свойства.

В этой ситуации разумно из соображений эффективности и ясности попытаться моделировать лишь одну из двух подсистем. Но тогда прошлое состояние клетки может быть запомнено в другой подрешетке, а не в другой плоскости битов. На "одном шаге четные позиции (т. е. те, где число $i+$у четно) представляют *настоящее,* а нечетные *прошлое.* Во время самого шага прошлое модифицируется правилом т и превращается в *будущее.* На следующем шаге они вновь меняются ролями.

Заметим, что на каждом шаге мы *просматриваем* лишь одну подрешетку и используем эту информацию только для *изменения* состояния другой. Ситуация такова, что если мы захотим вернуться назад во времени, то та же самая информация, которая была использована для изменения состояния, теперь доступна для "отмены" этого изменения (для обратимости изменение должно быть, конечно, *перестановкой* состояний клетки).

Таким образом, чередующиеся подрешетки могут рассматриваться как разновидность метода второго порядка, предназначенная для достижения обратимости. Этот подход (имеющий давние традиции в численном анализе) будет использован для анализа спиновых моделей Изинга в гл. 17.

14.4. Метод защитного контекста

В методе чередующихся подрешеток пространственное *положение* клетки используется для того, чтобы определить, может ли клетка быть обновлена на данном шаге без риска *потери информации* (это именно та тема, которой посвящено понятие обратимости). Можно придумать правила, в которых до-

статочная для этой цели информация закодирована в *состояниях* клеток; т. е. *контекст,* представленный совокупностью состояний соседей, используется для защиты от необратимых изменений. Только клетки, находящиеся в особом положении в окрестности с определенной конфигурацией состояний соседей, могут изменять свое состояние.

Этот метод, обсуждавшийся в [55,62], неудобен для использования и упомянут здесь только для полноты изложения.

14.5. Метод разбиения

Обратимость систем второго порядка, вроде обсуждавшихся в предыдущем разделе, выглядит несколько неожиданно. Отметим, что в (14.2) функция г может быть выбрана произвольно; если с являются вещественными числами, то *т* может использовать, скажем, возведение в квадрат и округление - операции, которые "отбрасывают" часть информации. Несмотря на это, полученная в результате динамика обратима, что означает *отсутствие потери информации.* Создается впечатление, что проведен сеанс математической магии.

Теперь мы обсудим метод порождения обратимых правил более целенаправленным и очевидным способом, а именно рассматривая *клеточные автоматы на разбиении,* некоторые примеры которых приводились в гл. 10 и 12.

В обычных клеточных автоматах устройство, которое порождает за один шаг новую конфигурацию из текущей, можно представить как массив *логических вентилей,* по одному на клетку, единообразно расположенных в пространстве и времени. На схеме, приведенной на рис. 14.1а (где для ясности представлено только одно пространственное измерение), каждый вентиль имеет несколько входных линий, соответствующих соседям клетки, но лишь одну выходную линию, в которой возникнет новое состояние клетки. В общем случае функция, вычисляемая вентилем этого вида, не может быть инвертируемой, поскольку лишь часть информации, имеющейся на входах, может появиться на выходе, а часть информации будет *утеряна.*[1] Верно, конечно, что некоторая часть входной информации каждого вентиля просматривается и соседними вентилями, и поэтому существует возможность, что потерянное в данном месте может сохраниться в каком-то виде где-либо еще.

[1] Эта потеря информации не может быть приписана тому, что данная функция делает при каком-то *конкретном* назначении входов; необходимо принять во внимание все возможные назначения входов и посмотреть, является ли соответствие вход-выход таковым, что по выходу можно всегда восстановить вход.

Однако чтобы гарантировать, что никакая часть информации не потеряется где-нибудь во время работы клеточного автомата, необходимы очень хитроумные предписания.

По приведенным выше причинам почти всякое правило, которое можно записать, дает *необратимый* клеточный автомат (действительно, еще несколько лет назад была известна только горстка обратимых правил, но и те были чрезвычайно тривиальны [58]).

В клеточных автоматах на разбиении, с другой стороны, благодаря специальной дисциплине информационного потока существует непосредственная связь между обратимостью всей динамической системы и обратимостью отдельных вентилей, которые образуют массив. Поэтому обратимость может быть прямо запрограммирована, что иллюстрируется использованием окрестности Марголуса (см. гл.12), одной из простейших схем разбиения.

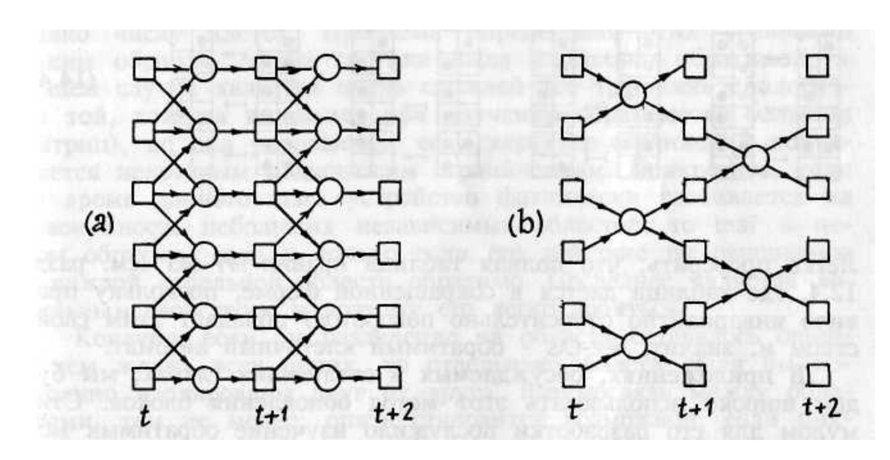
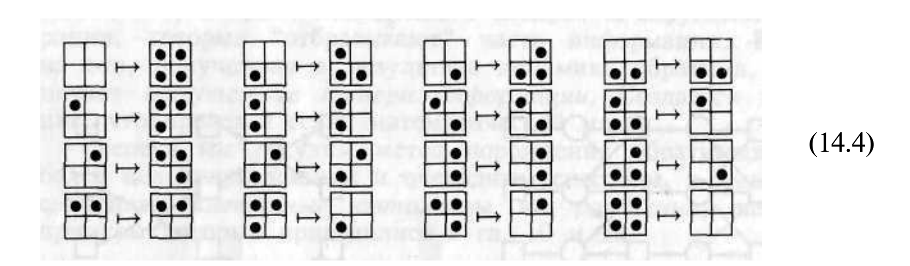


Рис. 14.1. (а) В обычном клеточном автомате вентиль связан с каждой *клеткой* и имеет много входов, но лишь один выход. (Ь) В клеточном автомате на разбиении вентиль связан с каждым *блоком* и имеет столько же выходов, сколько и входов.

В этой окрестности правило использует в качестве входов четыре клетки блока и возвращает в качестве выходов новые состояния *всех четырех клеток* того же блока. Таким образом, структура клеточного автомата может быть представлена как однородный массив вентилей *с одинаковым количеством входов и выходов* у каждого (рис. 14.1b), где ни один из входов не является общим для соседних вентилей. В этой ситуации каждый вентиль просматривает информационный поток

для всего блока из четырех клеток и осуществляет общий контроль над ним. Если отдельный вентиль обратим, то глобальный процесс также обратим. Если вентиль теряет информацию, то ни один из его соседей не будет в состоянии восполнить эти потери, и процесс заведомо будет необратим.

Правило для окрестности Марголуса будет обратимо, если и только если оно устанавливает взаимно-однозначное соответствие между старым и новым состояниями отдельного блока. Другими словами, если в первом столбце справочной таблицы, определяющей данное правило, мы приводим список всех возможных состояний блока, то, чтобы правило было обратимо, второй столбец должен быть перестановкой первого, как в следующем примере (заметим, что каждое из шестнадцати состояний левого столбца появляется где-нибудь в правом столбце);



(14.4)

Легко проверить, что полная таблица правил HPP-SAS (см. разд. 12.4, где таблица дается в сокращенной форме, поскольку правило инвариантно относительно поворотов) обладает этим свойством и, значит, HPP-GAS - обратимый клеточный автомат.

В приложениях, обсуждаемых в следующих главах, мы будем широко использовать этот метод обновления блоков. Стимулом для его разработки послужило изучение обратимых моделей вычислений, а в действительности его первым применением была реализация на клеточном автомате "компьютеров из биллиардных шаров" (см. гл.18).

Как вернуться назад во времени, сконструировав обратимое правило по методом разбиения и выполнив несколько его шагов? Очевидно, выполняя обратимые шаги в обратном порядке. Предположим для определенности, что мы воспользовались правилом таблицы (14.4) и что последний шаг был сделан на нечетной решетке. Тогда первый шаг обратного процесса будет *опять* использовать нечетную решетку (мы хотим отменить то, что только что было сделано для каждого блока) и будет применять правило, полученное из таблицы (14.4) обращением направления стрелок, т. е. поиском аргументов во

втором столбце и нахождением соответствующих результатов в *первом*. Фактически эта новая таблица определяет перестановку, являющуюся обращением исходной. Мы продолжим использование обратимого правила при движении в обратном направлении, чередуя как и прежде, четные и нечетные решетки. Для такого правила, как HPP-GAS (см. разд. 12.3), которое, к счастью, обратно самому себе, все, что нужно сделать для начала движения в обратном направлении - использовать дважды одну и ту же решетку и затем придерживаться обычного чередования.

Это обсуждение может быть легко обобщено на большее число измерений, состояний клетки и более сложные изменения правил и схем разбиения.[1] Основные рассуждения могут быть подытожены следующим образом.

Клеточный автомат можно представлять себе как распределенное в пространстве устройство, эволюция которого управляется системой взаимосвязанных уравнений; число уравнений равно числу клеток. Проблема определения этих уравнений таким образом, чтобы система была *глобально* обратимой, в общем случае является очень сложной (ее трудность аналогична той, которая возникает при изучении обратимости больших матриц), но она упрощается, если характер соединений подчиняется некоторым подходящим ограничениям. Конкретнее, если за время данного шага устройство фактически разбивается на совокупность небольших независимых областей, то шаг в целом обратим, если и только если его действие по отношению к каждой отдельной области обратимо. Последнее является *локальным* свойством, и поэтому его легко добиться.

Конечно, если бы разделение на области оставалось одним и тем же на всех шагах, то отдельные области были бы постоянно изолированы друг от друга. Взаимосвязь между областями, тем не менее, опять становится возможной, если циклически чередовать при последовательных шагах различные схемы разбиения. При этом способе не только гарантируется обратимость, но также автоматически получается закон обратного движения. А именно:

> Чтобы отменить шаг, отмените преобразование каждой отдельной области.
>
> Чтобы отменить всю эволюцию во времени, отмените отдельные шаги от последнего до первого.

[1] См. правило TM-GAS разд. 12.7, где две различные версии того же основного рецепта используются для четных и нечетных шагов.

14.6. Обратимость *h* случайность

Если заполнить клетки нашего экрана случайно выбранными двоичными значениями, а затем позволить этой конфигурации эволюционировать по правилу LIFE, то мы увидим сложное переплетение структуры и движения (гл. 3) с различными уровнями организации. Если вместо LIFE мы проследим некоторую обратимую во времени эволюцию, то неизменно обнаружим, что на каждом шаге система выглядит такой же случайной, как и вначале; на физическом языке, энтропия не будет уменьшаться (но может - и в общем случае будет - увеличиваться, пока не станет настолько большой, насколько это возможно).

Это свойство обратимых систем можно вывести из простых соображений о числе состояний системы: так как *большинство* конфигураций выглядят случайными, то лишь очень незначительная часть случайных конфигураций может быть отображена за любое заданное число шагов в небольшое число просто выглядящих конфигураций, если отображение является обратимым. То есть не существует способа установить взаимно-однозначное соответствие между большим и малым множествами конфигураций.

Не следует делать вывод, что обратимые клеточные автоматы менее интересны, чем необратимые. Это всего лишь означает, что в этих системах, как и в физике, ничего особенно интересного не может возникнуть, если начать с максимально неупорядоченного состояния.

Глава 15

ДИФФУЗИЯ И РАВНОВЕСИЕ

Что, уносясь в пустоте,
в направлении книзу отвесном
Собственным весом тела
изначальные в некое время
В месте, неведомом нам,
начинают слегка отклоняться,
Так что едва и назвать
отклонением это возможно.

[Лукреций]

Прежде чем обсуждать гидродинамические модели и другие физические явления, в основе которых лежит статистическая механика, в этой главе мы представим простые модели диффузии и равновесия.

15.1. Управляемая шумом диффузия

В разд. 10 мы привели простую модель одномерного случайного блуждания. Система состояла из единственной частицы, которая на каждом шаге должна была двигаться вправо или влево в зависимости от исхода бросания монеты. Эту модель нельзя непосредственно обобщить на случай более чем одной частицы: две частицы могли бы попытаться занять одно и то же место.

Мы обсудили также физически более реалистичную модель. На каждом шаге система разбивалась на двухклеточные блоки и содержимое каждого блока тасовалось случайным образом. В этом простом случае тасовка могла иметь лишь один из двух исходов, а именно (1) оставить обе клетки блока такими, как они есть, или (2) переставить их содержимое. Разбиение на блоки менялось на каждом шаге, так что информация могла перемещаться от одного места к другому.

Средства программирования CAM, которые мы должны были синтезировать, чтобы ввести регламент разбиения на блоки, полезны и в значительно более общих ситуациях; по этой причине они предоставляются CAM как примитивы - в виде окрестности Марголуса. Здесь мы сперва займемся аналогичной проблемой *двумерной* диффузии, непосредственно используя эти средства.

Двумерное тасование. Клетка в блоке окрестности Марголуса, именуемая CENTER, которая должна быть заменена, автоматически соединяется с ее соседями по блоку, именуемыми по отношению к самой клетке как CLOCKWISE (по часовой стрелке), COUNTER-CLOCKWISE (против часовой стрелки) и OPPOSITE (напротив). Содержимое такого блока может быть перетасовано 4! различными способами; для наших целей будет достаточен выбор из двух различных тасований, а именно (1) *повернуть* содержимое блока на одну четверть оборота *по часовой стрелке* или (2) повернуть его на столько же *против часовой стрелки*. Напомним, что правило, которое должно вращать все блоки плоскости 0 по часовой стрелке, это просто

```
: CLOCKWISE
    CCW >PLNO ;
```

(см. разд. 12.7).

В данной модели диффузии решение, в каком направлении вращать блок, будет зависеть от исхода бросания монеты, представленного "шумящим соседом" RAND, а правило диффузии будет следующим:

```
: 2D-BROWNIAN
    RAND { CCW CW } >PLNO ;
```

Конечно, для того чтобы чередовать на последовательных шагах две решетки, должен действовать рабочий цикл ALT-GRID из разд. 12.6.

Бросание монеты для каждого блока. Указанное выше правило применяется в CAM отдельно к каждой клетке массива и, в частности, к четырем клеткам каждого блока; значение, возвращаемое RAND, может изменяться от блока к блоку и от шага к шагу, но на каждом шаге должно быть *одним и тем же* для всех четырех клеток любого блока. Поэтому генератору случайных чисел в плоскости 1 должна быть дана команда позволить всем четырем клеткам блока видеть один и тот же результат.

По аналогии с разд. 10.2 мешалкой в плоскости 1 будет правило

```
: MAR6-STIR
    CENTER' CW' OPP' CCW
        AND XOR XOR >PLN1 ;
```

Это правило предоставляет для каждой клетки случайный бит и, следовательно, четыре бита для каждого блока.[1] Эти четыре

[1] Как упомянуто в разд. 8.4, более высококачественная случайность может быть получена от аппаратного генератора случайных чисел или от другого модуля CAM.

бита объединяются симметричным образом в единственный случайный бит правилом

```
                    : RAND (-- 0|1)
      CENTER' CW OPP' CCW
              XOR XOR XOR ;
```

так что результат будет один и тот же, если смотреть из любой клетки одного блока (см. табл. (12.6)). Как обычно, плоскость 1 будет заполнена случайными зародышами.

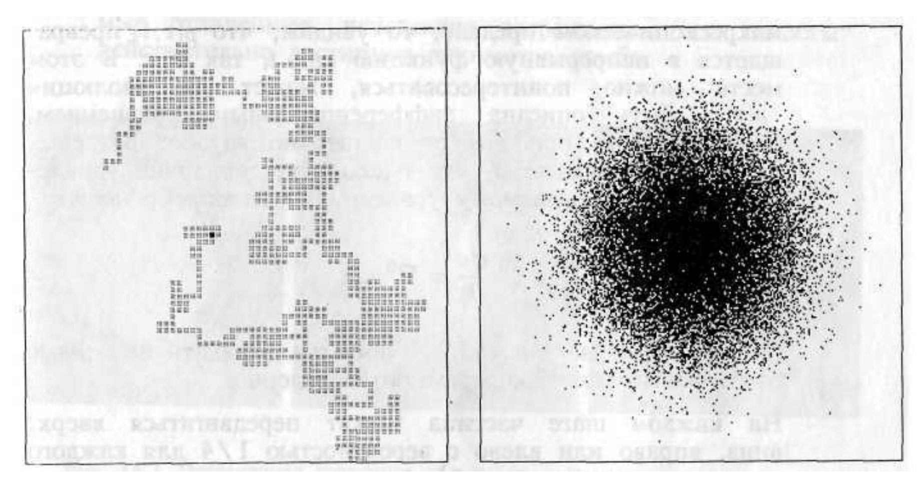


Рис. 15.1. (а) Запутанный путь частицы при двумерном случайном блуждании (увеличенное изображение). (Ь) Постепенная диффузия плотного кластера частиц.

Мы готовы к движению. Единственная частица, помещенная в плоскость 0, будет передвигаться с места на место резкими толчками, описывая запутанный, непредсказуемый путь (как на рис. 15.1а, где плоскость 2 была использована для записи следа частицы). Плотный кластер частиц, помещенный в середину экрана, будет медленно диффундировать во всех направлениях, как на рис. 15.1Ь; сравните этот результат с показанным на рис. 9.3а, который был получен при значительно более наивном подходе. Такое поведение имеет две замечательные особенности.

- Распределение частиц по всей области поперечником, скажем, двадцать клеток, совершенно равномерно; с некоторого расстояния, на котором отдельные частицы не могут быть больше ясно различимы, картина смотрится как полутоновое изображение: различные концентрации частиц кажутся разными оттенками серого цвета. Это

становится еще более заметным на рис. 15.2, где использовалась короткая временная выдержка (в этом случае сглаживание усреднением по времени эквивалентно сглаживанию усреднением по пространству). Имеет смысл связать с каждой точкой г изображения *плотность* *p(г)* окружающей области; по мере того как разрешение уменьшается, эта плотность становится непрерывной функцией г: из *битов* возникают *вещественные числа.*

Если мы рассмотрим *p* как функцию *времени* в том же макроскопическом пределе, то увидим, что *p(r,t)* превращается в непрерывную функцию как *t,* так и г. В этом месте можно поинтересоваться, может ли эволюция p(r, *t)* быть описана дифференциальным уравнением. Действительно, при выборе достаточно грубого разрешения будет обнаружено, что *Р* сколь угодно точно удовлетворяет так называемому уравнению *теплопроводности*

Серия кадров на рис. 15.2 - попытка передать некоторые особенности этого непрерывного поведения.

На каждом шаге частица может передвигаться вверх, вниз, вправо или влево с вероятностью 1/4 для каждого из четырех направлений[1]; промежуточные направления не допускаются. Поэтому на микроскопическом уровне правило, конечно, *неизотропно:* если бы можно было видеть только частицу, то тем не менее возможно было бы сделать вывод об ориентации осей решетки. Однако на макроскопическом уровне все направления равновероятны, и паттерн диффузии совершенно круглый. Это находится в поразительном контрасте с правилами клеточного автомата, которые мы представляли до сих пор, в которых ориентация решетки налагает свой отпечаток на эволюцию, независимо от того, насколько крупный масштаб используется.

Можно возразить, что "следом" светового конуса такого правила является *ромб,* подобный тому, который изображен на рис. 5.1: по прошествии времени *t* вероятность того, что частица находится в некотором заданном месте внутри ромба, будет отлична от нуля, тогда как вне ромба она будет точно нуль. По мере того как *t* увели-

[1] Можно игнорировать мелкозернистые корреляции, которые возникают из-за разбиения на блоки.

чивается, ромб будет пропорционально расти в размере, всегда сохраняя ту же *форму*. Будет ли эта форма распознаваема макроскопическим наблюдателем? Оказывается, что при $t \to \infty$ вероятность, что частица будет обнаружена на расстоянии много большем, чем $t^{1/2}$ от ее исходной позиции, становится пренебрежимо малой: распределение вероятности распространяется *значительно медленней,* чем световой конус, и большая его часть сохраняет круговую симметрию (см. рис. 15.6) - хотя самые отдаленные, исчезающе разреженные его "хвосты" действительно достигают периметра ромба.

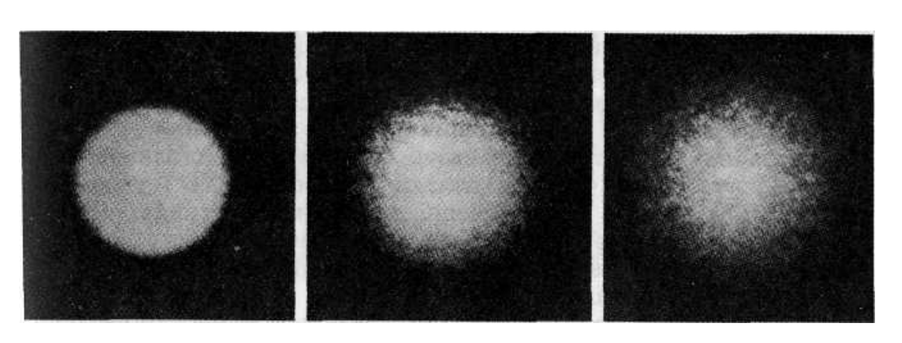


Рис. 15.2. Постепенная эволюция во времени плотности частиц *p*.

В заключение, в этой системе на макроскопической уровне возникает симметрия, которая не была представлена на микроскопическом уровне: забывается не только *шаг,* но и *ориентация* решетки. Первое из этих свойств является общим практически для всех решеточных моделей (и, значит, ничем не примечательно), в то время как второе оказывается очень редким и особенно желательным в контексте физического моделирования, поскольку на всех известных уровнях физика изотропна.

15.2. Расширение и установление теплового равновесия

В предыдущем примере генератор случайных чисел можно было себе представить как источник *теплового возбуждения*. В этой интерпретации один шаг модели на клеточном автомате соответствует физическому интервалу времени, длительность которого достаточна для того, чтобы настоящие молекулы мог-

ли пройти через бесчисленные столкновения в более мелком масштабе, чем тот, который представлен в модели, так что направление движения молекулы в конце интервала абсолютно не связано с направлением ее движения в начале: у молекулы было достаточно времени, чтобы, так сказать, забыть, куда она двигалась. В этой идеализации путь частицы чернил полностью определяется шумом.

А теперь мы рассмотрим противоположную идеализацию, а именно: (а) нет никакого внешнего шума, (Ь) частицы перемещаются в вакууме, и поэтому (с) всякие отклонения от прямолинейного пути появляются исключительно благодаря столкновениям с другими частицами.

Правило, подходящее для этой ситуации, это рассмотренное в разд. 12.7 правило TM-GAS. Здесь частицы перемещаются горизонтально или вертикально; две частицы, перемещающиеся в противоположных направлениях в двух смежных строках или столбцах, могут претерпевать "скользящее" столкновение. В этом случае обе частицы делают поворот на 90°, уходя из точки столкновения в противоположных направлениях. (Большинство соображений этого раздела применимо с тем же успехом к правилу HPP-GAS разд. 12.3, где, однако, частицы передвигаются в диагональных направлениях). Равномерное движение было достигнуто чередованием вращений содержимого блока окрестности Марголуса на последовательных шагах по часовой стрелке или против часовой стрелки - в неизменной предопределенной последовательности, а не по прихоти генератора шума (см. в конце разд. 10.2 описание аналогичного поведения в одном измерении). Столкновения достигались замещением шага вращения шагом "без изменений" всякий раз, когда две частицы занимали противоположные углы блока. Мы напомним здесь правило

```
                              : TM-GAS
               COLLISION IF
               CENTERS ELSE
      PHASE { CCW CW } THEN
                    >PLNO ;
```

Для нашего первого эксперимента мы поместим этот газ в "бутылку". Контуры этого сосуда (рис. 15.3а) будут нарисованы в плоскости битов 1. Столкновения между частицами и стенкой сосуда будут учитываться за счет следующей модификации правила. Если какая-то часть стенки попадает внутрь данного блока, то вращение содержимого блока на этом шаге не выполняется - так же как и в случае столкновения частицы с частицей.

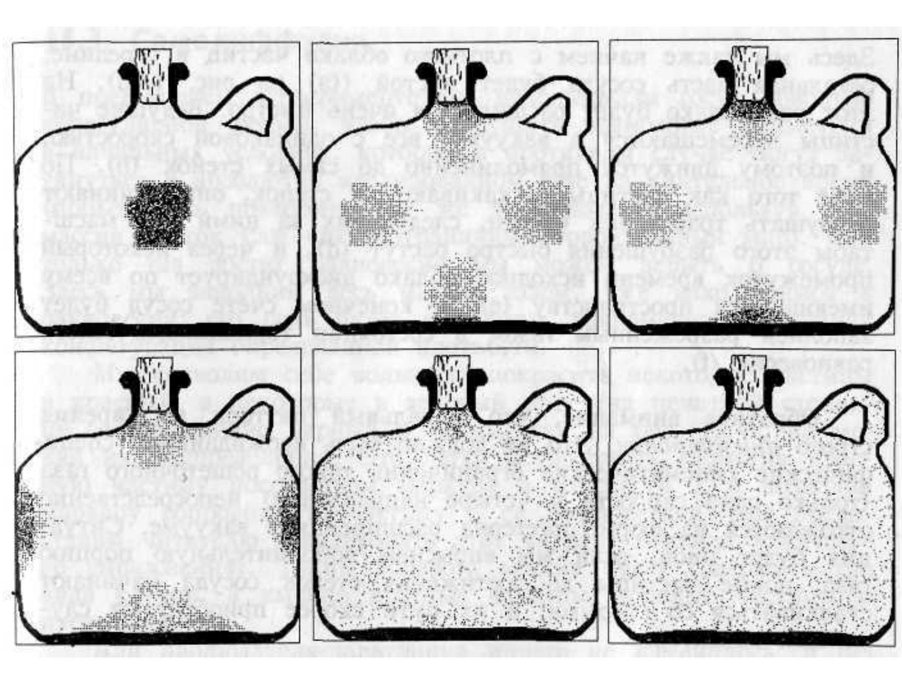


Рис. 15.3. Расширение облака TM-GAS в вакууме. Многократные столкновения между частицами и стенками сосуда в конечном счете приводят к установлению теплового равновесия.

Читатель может проверить, что согласно этому условию частицы действительно отскакивают от стенки.[1] Правило отскакивания следующее:

```
                          : WALL ( -- 0|1)
CENTER' CW' OPP' CCW'
            OR OR OR ;
                          : TM-GAS/WALLS
 COLLISION WALL OR IF
         CENTER ELSE
PHASE { CCW CW } THEN
               >PLNO
      CENTER' >PLN1 ;    \ стенки не двигаются
```

[1] Отражения от стенок не будут зеркальными - но, конечно, на уровне молекулы стенки не могут быть представлены "гладкими" или имеющими определенную ориентацию.

Здесь мы также начнем с плотного облака частиц в середине; остальная часть сосуда будет пустой ((а) на рис. 15.3). На этот раз облако будет расширяться очень быстро. Ведущие частицы перемещаются в вакууме, все с одинаковой скоростью, и поэтому движутся прямолинейно до самых стенок (Ь). По мере того как частицы отскакивают от стенок, они начинают разрушать траектории частиц, следующих за ними (с); масштабы этого разрушения быстро растут (d), и через некоторый промежуток времени исходное облако диффундирует по всему имеющемуся пространству (е). В конечном счете сосуд будет заполнен разреженным газом в состоянии термодинамического равновесия (f).

Обратите внимание, что начальный паттерн расширения строго *анизотропен.* Это не должно быть неожиданным: специфические кинематические ограничения такого решеточного газа (только одна скорость и четыре направления) непосредственно отражаются на форме паттерна расширения в вакууме. Ситуация будет иной, если мы впрыснем дополнительную порцию газа: новые частицы до достижения стенок сосуда начинают сталкиваться со старыми, а их пути скорее приобретают случайный характер.

Давайте впрыскивать все больше и больше газа до тех пор, пока не будет занята ощутимая доля клеток. В состоянии равновесия средняя длина пробега составляет теперь лишь несколько клеток (см. разд. 15.4) и отдельная частица будет следовать запутанным путем, очень похожим на путь из предыдущего эксперимента. Тепловое возбуждение теперь ведется *всей совокупностью сохраняющихся частиц газа,* а не внешним источником шума.

То обстоятельство, что система как замкнутое целое полностью детерминирована, почти не меняет ситуацию для отдельной частицы, по-прежнему плавающей в море шума (см. разделы 17.3-17.5). Поэтому вопрос о том, является ли некоторое правило клеточного автомата *детерминированным* или *стохастическим,* непринципиален: *клинамен* Лукреция[1] - это лишнее понятие, для объяснения физики процесса оно не требуется.

[1] В своей книге *"О природе вещей"* римский поэт Лукреций (1-е столетие до н. э.) дает бесстыдно редукционистскую - и поразительно современно звучащую - картину физического мира. *Кпинаменом* он назвал чрезвычайно слабое случайное отклонение атомов от их прямых путей, которое автор считал необходимым ввести (несколько неохотно), чтобы заставить его модель делать то, что, как он считал, она должна делать, но не могла без этой поправки.

15.3. Самодиффузия

В предыдущем эксперименте мы использовали сосуд довольно нерегулярной формы, чтобы разрушить возможные симметрии в начальных условиях, которые в противном случае могут затянуться в абсолютно изолированном газе на длительное время (попытайтесь повторить тот же эксперимент, используя весь массив клеток САМ как безграничный тороидальный мир). Как только газ достиг равновесия мы можем забыть о сосуде. Здесь мы заполним весь массив газом, уже находящимся в равновесии; его роль будет выполнять равномерная случайная конфигурация определенной плотности.

Мы позволим себе вольность покрасить некоторые частицы в красный, а некоторые в зеленый цвет (на печатной странице соответственно черный и серый, как на рис. 15.4), не меняя их распределения. Это легко осуществить, пометив красную частицу посредством маркера в плоскости 1 и позволив маркеру сопровождать частицу в ее путешествии (мы предоставим читателю закодировать этот вариант правила в качестве упражнения). Таким образом, массив будет содержать три вида клеток, а именно красные частицы, зеленые частицы и пустые клетки.

Для наблюдателя-дальтоника ничего не изменилось, и газ по-прежнему в равновесии; однако чувствительный к цвету наблюдатель будет видеть красный и зеленый газы, диффундирующие друг в друга (рис. 15.4), до тех пор, пока эта нововведенная степень свободы не достигнет равновесия (в виде желтой смеси). Цвет не влияет на динамику и по этой причине помогает нам понять, что даже в "неокрашенном" газе *самодиффузия* является реальным явлением; скорость самодиффузии является одним из параметров переноса, которые характеризуют отклик газа на возмущения равновесия (см. гл. 16).

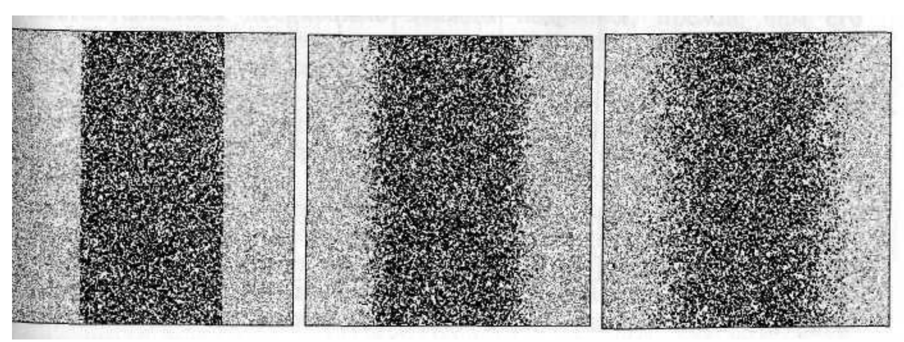


Рис. 15.4. Самодиффузия двух газов, имеющих различный цвет, но идентичные динамические свойства.

' 1" • ,*

15.4. Средняя длина свободного пробега

Для иллюстрации вычислим среднюю длину свободного пробега частицы как функцию плотности частиц для модели TM-GAS. С точки зрения частицы остальные три клетки ее блока в каждый момент будут заняты с вероятностями, заданными следующей таблицей (где сплошной кружочек обозначает рассматриваемую частицу, а является плотностью частиц)

●			●	o	
				o	
●	o		●		
			o	o	
●			●	o	$P^2(l-p)$
o				o	
●			●	o	„3
	o	$p(1-p)^2$	o	o	

(15.1)

Согласно четвертому элементу этой таблицы, вероятность столкновения (при этом мы имеем в виду *бинарное* столкновение - единственный вид столкновений, которые могут отклонять траекторию частицы в TM-GAS) есть $p = p(\backslash-p)^2$. Можно легко выразить вероятность некоторой последовательности столкновений; например, вероятность, что за семь шагов частица столкнется, скажем, на втором и седьмом шагах, равна

$$qpqqqqp = q^5 p^2$$

(где q = 1-p), а *свободный пробег* длины / (т.е. свободный полет в течение / шагов, за которыми следует столкновение на (й-1)-м шаге) возникнет с вероятностью $q'p$.

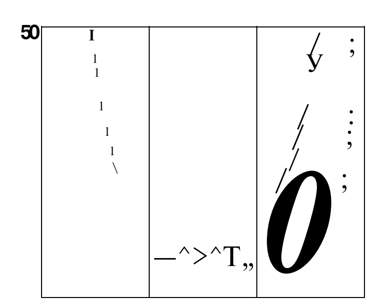


Рис. 15.5. Сплошная линия - зависимость средней длины свободного пробега Л от плотности частиц p в TM-GAS; штриховая линия - зависимость средней длины свободного пробега для "дырок".

Средняя длина свободного пробега, т. е. взвешенная сумма всех возможных длин свободного пробега, поэтому равна

$$Л = Op + lqp + 2qqp + Zqqqp \text{ H————————} \quad (15.2)$$

$$= qp\{\backslash + 2q + 3{<}z^2 + \cdot\cdot\cdot)$$

$$\frac{qp}{(l\text{-}q)^2} \quad \hat{} \quad \frac{q}{p}$$

Можно получить тот же результат более простыми рассуждениями. Если частица имеет вероятность *p* претерпеть столкновение на любом шаге, то в среднем она будет затрачивать время *1/p* от одного столкновения до следующего (среднее *время* свободного пробега); так как в течение одной единицы этого времени частица "зафиксирована" (в TM-GAS во время столкновения перемещение не происходит), то средняя *длина* свободного пробега равна 1/p - 1 = *q/p*.

На рис. 15.5 приведен график зависимости средней длины свободного пробега частицы Л от плотности частиц *P* . Эта кривая имеет минимум Л = 23/4 (»6) при *p* = 1/3 . Для малых значений *p,* частицы перемещаются в вакууме почти свободно; при высоких плотностях они скользят почти свободно одна мимо другой (поскольку встречи более чем двух частиц запрещают столкновения).

Так как правило симметрично по отношению к частицам и вакууму (единицы и нули), то "дырки" в море частиц ведут себя во многом так же, как частицы в море вакуума.

15.5. Проявление изобретательности

Здесь мы опишем эксперимент такого сорта, что в обычных обстоятельствах необходимо дважды подумать, прежде чем его предпринять, и к которому вместо этого можно с легким сердцем приступить на машине клеточных автоматов.

Когда диффузия управляется внешним источником случайности, каждый отрезок пути частицы совершенно независим от предыдущего: шум на каждом шаге абсолютно свежий. Однако, когда путь частицы определяется столкновениями с другими частицами, могут возникать корреляции; интуитивно, некоторые из результатов одного столкновения могут "обойти кругом", чтобы повлиять на ту же частицу. Так как предполагается, что отклонения от идеального случайного блуждания незначительны, то необходимо длительное накопление статистических данных.

В настоящем эксперименте путь одной частицы газа, начинающийся в начале координат, прослеживается в течение фик-

сированного числа шагов *t* (скажем, нескольких тысяч). Путь частицы записывается и, в частности, отмечается ее конечная позиция. Процедура повторяется большое число и раз, каждый раз с другим начальным расположением частиц. Каждый прогон, конечно, влечет за собой моделирование всей газовой системы, состоящей из десятков тысяч частиц. Над собранными данными затем могут быть выполнены различные виды статистического анализа.

Этот эксперимент был проведен в MIT Андреа Калифано. Одним из результатов является численное определение распределения вероятности *P(x,y;t);* это показано на рис. 15.6 для <-1024 шагов. Для любого значения *x, y* высота кривой представляет действительное число прогонов, при котором путь частицы оканчивается в точке *x, y:* суммарное число *прогонов,* следовательно, равняется *объему* "гауссовской кучи" на рисунке. Для того чтобы получить приемлемое разрешение для низких значений *P,* задав широкий динамический диапазон, необходимо совершить очень большое число шагов п - в этом случае порядка полмиллиона. Таким образом, кривая рис. 15.6 соответствует вычислению, содержащему в сумме 256 x 256 x x 1024x500 000 (около 30 триллионов) обновлений клеток.

Кто будет записывать шаг за шагом позицию выбранной частицы? Если мы изобразим частицу красным, то глаз сможет относительно легко отслеживать ее траекторию на экране; однако координаты такой частицы нигде не появляются в явном виде при моделировании: мы имеем всего лишь массив клеток, некоторые из которых оказываются зелеными, одна красной, а остальные пустыми. Мы могли бы дать команду компьютеру-хозяину останавливать моделирование после каждого шага, считывать содержимое массива и искать клетку, содержащую красную частицу, но это было бы чрезвычайно неэффективно. Метод, который на самом деле использовался, состоит в следующем.

Предположим, что мы начали с PHASE - о и с частицей в левом верхнем углу блока окрестности Марголуса. Если условие COLLISION истинно, то частица на первом шаге не переместится; в противном случае она будет передвигаться вправо (так как весь блок поворачивается по часовой стрелке). В этой ситуации, чтобы узнать положение частицы в конце первого шага, мы не должны просматривать весь массив; нам необходимо задать лишь один бит информации, а именно: столкнулась частица или нет. На следующем шаге мы узнаем, что PHASE » 1, а из бита, взятого на предыдущем шаге, мы узнаем позицию частицы внутри блока в начале текущего шага. С другой стороны, единственный бит информации даст нам возможность определить новое положение частицы в конце этого шага. Поскольку мы интересуемся лишь этой частицей, то

полную историю моделирования в тысячу шагов бесспорно можно сжать в строку из тысячи битов. Компьютер-хозяин может легко обрабатывать один бит за шаг - точно так же, как САМ заменяет сотни тысяч битов в течение того же интервала времени.

Как объяснено более подробно в разд. 17.6, наряду со справочной таблицей, которая вычисляет новое состояние клетки *{функция перехода),* программируется вспомогательная справочная таблица с теми же данными об окрестности, чтобы вычислить последующие значения *(выходная функция).*

1, Если клетка содержит красную частицу и эта частица претерпевает столкновение.

О В противном случае.

Это значение загружается в счетчик событий, который накапливает события в клетках в ходе прогона; так как имеется только одна красная клетка, то суммарное количество в конце каждого шага будет 0 или 1; это вся информация, которая нам необходима, и она может быть считана со счетчика событий в конце каждого шага без нарушения моделирования.

Для многих исследовательских целей такая строка битов может быть обработана в реальном времени, по мере того как она появляется из машины. Однако сама строка охватывает все относящиеся к эксперименту данные в такой компактной форме, что сохранение ее, скажем, в файле на диске, не представляет проблем. Таким же образом одна и та же строка может быть использована и для многих других исследовательских целей. Например:

- "Объединением" строки отдельного шага можно восстановить конечную позицию частицы на этом шаге и, таким образом, добавить еще один объемный элемент в гистограмму, такую как на рис. 15.6.

- Строка может быть сопоставлена с той, которая порождена идеальным генератором случайных чисел.

- Можно подсчитать корреляцию строки с версией этой же строки, взятой с некоторым запаздыванием, это, вероятно, наиболее существенная часть исследования в эксперименте данного вида.

Методы вычисления автокорреляций, включающие все содержимое плоскостей битов - а не единственный бит, извлеченный указанным выше способом, будут обсуждаться в разд. 16.6.

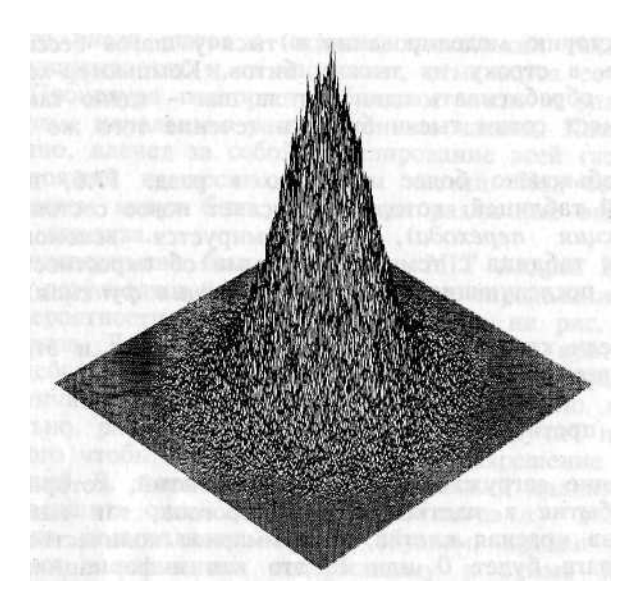


Рис. 15.6. Гистограмма функции $P(x, y;$ O - вероятности того, что частица TM-GAS будет обнаружена в момент времени t в положении $x, y,$ полученная продолжительной серией шагов моделирования на CAM.

15.6. Регулируемый источник шума

Решеточные газы, такие как HPP-GAS и TM-GAS, дают удобные источники шума для экспериментов на машине клеточных автоматов. Наиболее полезное их свойство состоит в том, что поскольку начальное число единиц сохраняется этими правилами, то вероятность p обнаружения единицы в любой позиции постоянна и регулируется в широком диапазоне малыми приращениями. Чтобы получить другую вероятность, не обязательно загружать новое "перемешивающее" или "отбирающее" правило: можно просто добавить или удалить частицы из плоскости битов.

Кроме того, равновесные свойства этих газов хорошо описаны. Спонтанное выравнивание плотности и давления, замеченное в разд. 15.2, аналогично тому, которое наблюдается у вязкой жидкости, и подчиняется аналогичным законам (см. разд. 16.2). Средняя длина свободного пробега, которая зависит от того, насколько интенсивным является процесс перемешивания, меньше дюжины клеток на большей части диапазона плотностей (см. рис. 15.5). Локальные корреляции, конечно, существуют, поскольку информация не может распространяться

быстрее, чем на одну клетку за шаг, но они весьма кратковременны и хорошо понятны (см. разд. 16.6).

Весьма полезно также наличие двух различных газов, в одном из которых частицы перемещаются по строкам, и столб-; цам (тм-GAS), а в другом - по диагонали (HPP-GAS). Например, очень маловероятно, что на правило, основанное преимущественно на горизонтальных и вертикальных обменах информации (т. е. то, которое использует окрестность фон Неймана), могут повлиять корреляции движущегося по диагонали газа.

Два газа могут быть запущены в различных плоскостях битов и использованы как объединенный источник шума, например, взятием операций AND и XOR от их содержимого. Одно преимущество состоит в том, что два источника шума, основанные на различных механизмах, в некоторой степени компенсируют присущие друг другу ограничения. Другое преимущество то, что взятием произведения вероятностей p и p', заданных двумя плоскостями битов, можно синтезировать намного меньшую вероятность pp без использования газов слишком низкой плотности (что дало бы длинный средний свободный пробег и, следовательно, медленное перемешивание).

Наконец, локальные корреляции могут быть практически устранены применением дополнительных модулей CAM. Действительно, архитектура CAM позволяет сдвигать пространственное начало одного модуля относительно другого в любом направлении на любую величину просто путем изменения содержимого регистра. Это делает возможным "воздействие на расстоянии" и, в частности, допускает глобальную (а не только локальную) перетасовку данных. Для этого достаточно запустить генератор шума на основе решеточного газа и на каждом шаге сдвигать начало координат этого модуля на случайную величину как по вертикали, так и по горизонтали; случайные числа для этого сдвига требуются с весьма низкой частотой одно или два за шаг, что легко обеспечивается компьютером-хозяином, использующим любой из ряда известных алгоритмов. Этот метод использовался для эксперимента, изображенного на рис. 17.6, где для моделирования идеального термостата потребовался высококачественный генератор случайных чисел.

15.7. Ограниченное диффузией агрегирование

Ограниченное диффузией агрегирование возникает, когда частицы налипают на начальный зародыш, представленный фиксированным объектом, и постепенно наращивают его. Как мы сейчас увидим, зародыш обычно дает нерегулярный дендритообразный рост, напоминающий морозные узоры на окне. Ограниченное диффузией агрегирование является приемлемой мо-

делью физических процессов роста (например, кристаллов льда), в которых рост дендритов возникает потому, что материал, необходимый для роста, должен диффундировать извне или же должен удаляться некоторый побочный продукт роста (скажем, тепло).

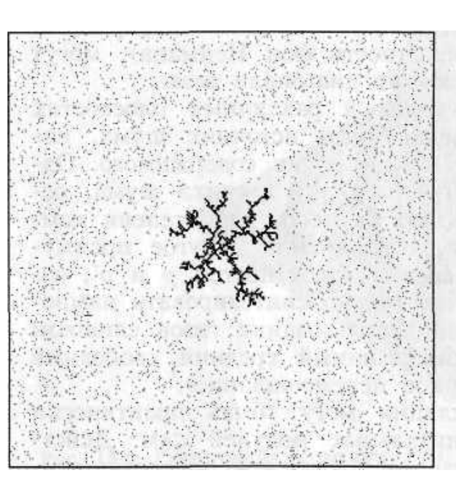
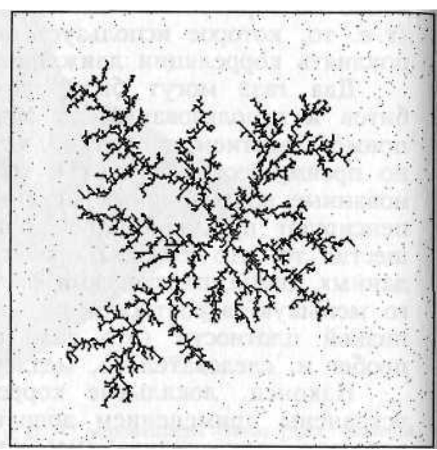



Рис. 15.7. Дендритный рост за счет ограниченного диффузией агрегирования. Процесс был начат с одноклеточного зародыша в середине при 10% плотности диффундирующих частиц.

Ниже приводится вариант правила, предложенного Чарлзом Беннеттом. В CAM-A ПЛОСКОСТЬ 0 будет содержать диффундирующие частицы, как и в правиле 2D-BROWNIAN разд. 15.1, а плоскость 1 будет содержать начальный зародыш, и в ней будет располагаться растущий дендрит. Обе плоскости используют окрестность Марголуса. Диффузия управляется генератором шума в CAM-B.

Механизм роста очень прост: наличие дендрита обнаруживается так же, как и стенка сосуда в разд. 15.2. Если часть дендрита появляется где-либо в блоке, то любые диффундирующие частицы, содержащиеся в этом блоке, будут "прилипать" к дендриту, т. е. передаваться из плоскости 0 в плоскость 1, где они сохраняются неподвижными. Результат напоминает действие липкой бумаги для мух.

```
CAM-A N/MARG &/CENTERS
                        RAND  (-- 0|1)
         &CENTER'        \ шум поступает из CAM-B
                        WALL  (-- 0|1)
```

```
      CENTER' CW' OPP' CCW'     \ считывать стенку
                OR OR OR ;
                           : DENDRITE
               WALL IF      \ Частицы: Если на стенке,
               O ELSE       \   то прилипнуть к ней
RAND { CW CCW } THEN >PLNO  \   иначе сохранить движение
               WALL IF      \ Дендрит: Если стенка, то
    CENTER' CENTER OR ELSE  \   захватывать любые частицы,
       CENTER' THEN >PLN1 ; \   иначе без изменений
```

Поскольку связь между CAM-A И CAM-B осуществляется только через центральную клетку, то для того чтобы обеспечить в CAM-в генератор случайных чисел, подходящий для окрестности Марголуса в CAM-A, необходима некоторая изобретательность. Читатель может пропустить следующее описание.

```
CAM-B N/MARG-HV
                    LE/DELAY
  UL  UR  XOR                 \ Взять данные из обеих
  LL  LR  AND  XOR            \   плоскостей, перемешать
  UL' LL' AND                 \   и возвратить тот же исход
  UR' LR' XOR XOR XOR >PLN2   \   во все четыре клетки
               CENTER >PLN3 ; \ Линия задержки на один шаг
```

В разд. 15.1 мы отметили, что для того, чтобы предпринять согласованное действие, все четыре клетки должны видеть один и тот же случайный исход; там правило RAND применяло операцию XOR К четырем соседям по блоку, чтобы выработать такой инвариантный по отношению к позиции в блоке исход. Здесь, поскольку CAM-A не может видеть четырех соседей по блоку в CAM-B, такой инвариантный исход вычисляется внутри CAM-B И *запоминается* в плоскости 2. Однако возникает проблема: этот исход доступен лишь шагом позже, когда машина уже использует *другое* разбиение на части. Чтобы получить его в подходящей фазе, мы задерживаем исход еще на один шаг, копируя его из плоскости 2 в плоскость 3, где CAM-A будет видеть его при помощи слова &CENTER'.

Чтобы дать приемлемый случайный результат, функция, которая должна вырабатывать позиционно-инвариантный исход для плоскости 2, не может состоять только из операций XOR; требуется также нелинейная компонента, такая как операция AND. Это разрушает вращательную симметрию функции и заставляет нас использовать абсолютных соседей по блоку UL, UR и т.д. (см. разд. 12.5).

Чтобы инициализировать генератор случайных чисел, поместите случайный паттерн в плоскость 2 и выполните два шага. Это заполнит "линию задержки" в один шаг достоверными данными, и вы сможете теперь продолжить инициализацию плоскостей CAM-A.

На рис. 15.7 изображены ранняя и поздняя стадии роста дендритов, полученные согласно этому правилу, начиная с

плотности частиц 10% (см. также фото 12). Образец имеет вид разветвленного кластера; ветви формируются, потому что прогрессирующее обеднение частицами внутренностей выемок существующего кластера подавляет рост там. Как бы то ни было, кончики ветвей стремятся первыми подхватывать диффундирующие частицы, и лишь немногие из них проникают в пространство между ветвями. Для более высоких плотностей диффундирующих частиц форма кластера агрегирования более глобулярна; для более низких плотностей более дендритообразна. В пределе малой плотности ограниченное диффузией агрегирование порождает паттерны, имеющие вполне определенную фрактальную размерность; в двумерных моделях фрактальная размерность близка к 1.7 (см. [53], стр. 121).

Глава 16

ДИНАМИКА ЖИДКОСТЕЙ

В этой главе рассматривается моделирование динамики жидкостей посредством сведения ее к микроскопическим законам (в несколько идеализированном виде).

В очень большой системе даже относительно небольшие ее части по-прежнему содержат большое число частиц и поэтому допускают осмысленный макроскопический анализ. Интересная ситуация возникает, когда равновесие само установилось на некотором масштабе, но не на более крупном, так что система в целом не находится в равновесии. В этой ситуации макроскопические параметры постепенно меняются от места к месту;[1] кроме того, в каждом месте эти параметры могут также постепенно меняться *во времени;* движущими силами такой макроскопической эволюции, конечно, будут пространственные градиенты макроскопических величин.

Систему такого типа можно представлять себе как совокупность локальных равновесий, которые плавно переходят друг в друга и эволюционируют во времени; различные макроскопические величины непрерывно изменяются.

16.1. Звуковые волны

Мы продолжим здесь серию экспериментов, начатых в гл. 15. Согласно такому правилу, как TM-GAS, частицы (представленные единицами в вакууме из нулей) перемещаются по прямым линиям и сталкиваются друг с другом; импульс при этих столкновениях сохраняется. Столкновения приводят к постепенной рандомизации путей частиц, и в конечном счете газ достигает равновесия.

Что произойдет, если мы вдруг нарушим это равновесие? Начнем для определенности с конфигурации, имеющей плотность р = 1/2, и заменим небольшой объем этого газа плотно заполненным облаком частиц (рис. 16.1а). Облако тут же начнет расширяться, сжимая окружающий газ; импульс этого стремительного движения наружу таков, что в определенный момент времени центр возмущения окажется обедненным частицами (Ь). В свою очередь кольцо сжатия будет также расширяться и наружу, и внутрь, создавая таким образом в центре новый пик давления. Чередующиеся сжатия и разрежения

[1] Изменения являются постепенными, потому что любые две части системы, которые существенно перекрываются, должны иметь почти идентичные параметры.

распространяются наружу как *звуковая волна* (с). Суммарный результат таков, как будто в пруд вылили ведро воды.

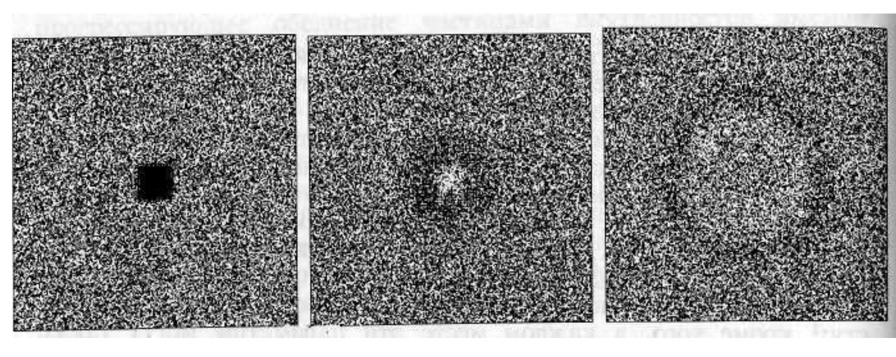


Рис. 16.1. Круговая волна, порожденная локализованным возмущением.

На фото 13 это явление показано при более высоком разрешении с использованием версии HPP-GAS для четырех плоскостей битов, описанной в разд. 16.5.

Имеется ряд удивительных фактов, относящихся к указанному выше поведению:

- На временном масштабе, сравнимом с размером возмущения, газ ведет себя как *упругая среда;* движение недостаточно демпфировано. Избыточная плотность в центре не просто медленно *вытекает* наружу; она *бросается* наружу; пытаясь выровнять давление, газ расширяется больше, чем следует, а когда пытается исправить первую попытку, снова уплотняется больше, чем необходимо.

- Хотя микроскопические законы строго анизотропны (существуют привилегированные направления движения, см. рис. 15.3Ь), звуковая волна является *круговой.* Подобно диффузии из разд. 15.1, здесь коллективное поведение частиц также проявляет симметрию, которой нет в поведении отдельной частицы.

- Скорость звука v в этой среде существенно меньше, чем скорость частиц; если принять последнюю за единицу, то $v = 1/\sqrt{2}$(см. разд. 16.7). Эта скорость независима от направления, длины волны, а также, как оказывается, от плотности решеточного газа. Таким образом, на макроскопическом уровне возникает четко выраженное механическое свойство, которое не имеет соответствующих причин на микроскопическом уровне.

96.2. Гидродинамика

Звуковые волны - лишь один аспект феноменологии жидкостей, на самом деле не представляющий большого интереса для гидродинамики; мы возвратимся к нему в конце этой главы в связи с экспериментами по волновой оптике (разд. 16.7).

Гидродинамика интересуется преимущественно ситуациями, где различные части жидкости движутся по отношению друг к другу и по отношению к твердым препятствиям со скоростями, много меньшими скорости звука. В этом пределе, если пренебречь внешними силами, такими как гравитация, любые различия в плотности выравниваются за пренебрежимо малое время, и жидкость может рассматриваться как *несжимаемая*. Даже когда применяются эти упрощения, феноменология жидкостей может быть чрезвычайно разнообразной. В зависимости от скорости основного течения, размера и формы препятствий, вязкости жидкости, можно получать ламинарное течение, вихри, турбулентность и т. д. (см. в [65] богатый набор иллюстраций).

В этой ситуации подходящей переменной является *скорость* V (вектор) течения в различных точках, а подходящим параметром - *вязкость* жидкости *у* . Поведение жидкости определяется уравнением *Навье-Стокса*

$$\frac{\wedge}{at} + (VV)V = -\frac{}{p}Vp + iA7^2V, \qquad (16.1)$$

где *р* - давление, а р - плотность (постоянная). Это *нелинейное* дифференциальное уравнение, и за исключением отдельных случаев, чтобы найти его решение для заданных начальных и граничных условий, необходимо прибегнуть к численным методам. Большая часть машинного времени во всем мире расходуется на моделирование гидродинамических задач.

Оказывается, что на макроскопическом уровне газы, описываемые правилом клеточного автомата, вроде HPP-GAS или тм-GAS, приближенно удовлетворяют уравнению Навье-Стокса [23], а похожее правило на гексагональной решетке, т. е. FHP-GAS, рассмотренное в разд. 16.5, *в точности* удовлетворяет этому уравнению [18]. Многие исследователи стали в последнее время интересоваться такими моделями динамики жидкости, которые имеют ряд привлекательных концептуальных особенностей и немало обещают для практики [28].

Достаточно детальное моделирование задач динамики жидкостей требует очень больших вычислительных ресурсов. На рис. 16.2 изображено моделирование потока после препятствия, проведенное Салемом и Вольфрамом [50] (похожие экспери-

менты были проведены Дюмьером и др. [14] примерно в то же время). Этот эксперимент был выполнен на Connection Machine, которая была запрограммирована как клеточный автомат для массива размером примерно 5000x5000 бит; правило является вариантом FHP-6AS; стрелки показывают величину и направление течения. Для того чтобы начертить стрелки, массив был разделен на области приблизительно из 25000 бит каждая; компьютерная программа помимо самого моделирования вычислила в явном виде число частиц в каждой области, движущихся в разных направлениях.

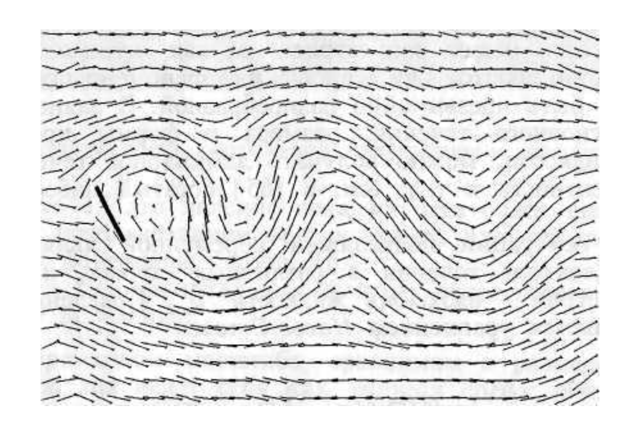


Рис. 16.2. Течение за препятствием (из работы Салема и Вольфрама).

163. Трассировка течения

Располагая одним модулем CAM, можно лишь еле-еле увидеть существенные аспекты гидродинамического течения, однако если взять большее число модулей или воспользоваться методом "вычерпывания", упомянутым в конце этого раздела, мы получим немалый диапазон возможностей. Для достаточно крупномасштабных экспериментов на должном уровне необходимы намного большие машины клеточных автоматов. Однако основные методы и понятия могут быть исследованы совершенно независимо от мощности вычислительной машины.

Состояние равновесия для TM-GAS, полученное случайным заполнением массива с определенной плотностью (скажем, 50%) частиц, не проявляет на макроскопическом уровне никакого суммарного течения: жидкость покоится. Чтобы вызвать медленный дрейф жидкости в каком-то направлении, мы должны увеличить долю частиц, двигающихся в этом направлении,

и уменьшить долю частиц, двигающихся в противоположном направлении. В отсутствие препятствий введенный таким образом импульс будет сохраняться.

Назовем PN, PS, pw, и PE "концентрациями" частиц, движущихся в четырех основных направлениях компаса, так что

$$P = PN + PS + PW + PE \qquad (16.2)$$

»

где *p* - суммарная плотность. В дрейфующем равновесном газе, даже если эти концентрации различны, каждая концентрация должна быть на больших временах *постоянной;* каждая популяция должна пополняться с той же скоростью, с какой она обедняется за счет столкновений.

При столкновении пара север /юг будет порождать пару запад/восток и наоборот; мы можем символически обозначить две эти "реакции" в виде

$$N + S \text{->} W + E \quad , \quad 'W + fi \text{->} N + S.$$

Скорость каждой реакции пропорциональна произведению концентраций сталкивающихся разновидностей частиц; соответствующие скорости, следовательно, равны

$$PNPS \quad , \quad PWPE- \qquad (16.3)$$

В равновесии две скорости реакции должны быть сбалансированы; поэтому равновесные концентрации должны удовлетворять соотношению

$$PNPS = \qquad (16.4)$$

которое вместе с (16.2) полностью характеризует состояние равновесия. Для малых скоростей дрейфа можно использовать линейную аппроксимацию

$$PN + Ps = Pw + PE,$$

где геометрические средние из (16.4) были заменены арифметическими средними.

Таким образом, если мы создадим начальную конфигурацию, где плотности частиц, движущихся на север или на юг, равны p/4, на восток (1 + 2e)p/4, и на запад (1 - 2e)p/4, то

газ будет близок к равновесию и дрейфовать как целое вправо со скоростью e . Такая конфигурация при $p = 1/2$ и $e = 1/10$ показана на рис. 16.3а. Поскольку плотность везде одинакова, то нет и макроскопических признаков движения газа; мы как бы наблюдаем за прозрачным течением без отличительных меток, за которыми можно было бы следить. Чтобы сделать линии тока видимыми, можно применять метод рис. 16.2, что, однако, требует большого объема нелокальной обработки. Здесь мы испытаем метод, который более близко напоминает то, что возможно в физическом эксперименте.

Идея состоит в том, чтобы поместить в течение несколько дискретных *трассеров* и проследить их траектории. Для этой цели мы пометим несколько частиц, как в разд. 15.3; движение каждой частицы будет, конечно, иметь случайную компоненту (броуновское движение), на которую наложится незначительное смещение вправо. На рис. 16.3Ь изображен дрейф помеченных частиц, испускаемых "дымовой трубой". Ясно, что на этом масштабе дрейф еле заметен и в целом замаскирован случайной компонентой.

Однако по мере того, как масштаб моделирования увеличивается, трассеры могут быть прослежены на протяжении более длительного времени, когерентная компонента движения растет как t, в то время как случайная компонента - лишь как $t^{1/2}$; по мере того как относительный вклад случайной компоненты уменьшается, начинают появляться отчетливые линии тока. На рис. 16.3с изображен тот же эксперимент, выполненный на массиве 1024x1024.

Вычерпывание. Хотя в эксперименте, изображенном на рис. 16.3, использовался массив, в шестнадцать раз больший чем в САМ, этот эксперимент был выполнен на единственном модуле САМ. Основная идея состоит в том, чтобы хранить большой массив в памяти компьютера-хозяина; вы берете небольшую "порцию" этого массива, передаете ее САМ ДЛЯ обработки, скажем на дюжину шагов, и снова запоминаете результат в большом массиве; затем вы переходите к Следующему зачерпыванию и т.д.[1] Когда проход по всему массиву завершен (весь массив продвинулся теперь вперед на дюжину шагов), выполняется новый проход; этот цикл можно продолжать сколько угодно раз.

[1] Поскольку порция внутри САМ не получает информацию с краев от смежных с ней порций, все еще находящихся в компьютере-хозяине, то после двенадцати шагов ее край будет содержать бесполезные данные толщиной в двенадцать клеток. Эта часть данных отбрасывается и, таким образом, каждая операция вычерпывания обновляет часть массива, которая чуть меньше, чем сама порция.

Используя такой метод, CAM может обновить массив произвольного размера; время обновления в пересчете на одну клетку увеличится при этом меньше чем в два раза.

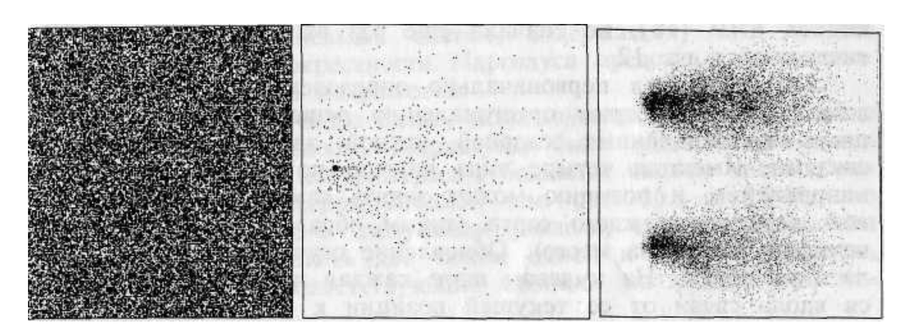


Рис. 16.3. (а) Направление дрейфа невидимо, если жидкость имеет везде одинаковую плотность. (Ь) Метки, испускаемые дымовой трубой, диффундируют в жидкость, (с) При большем масштабе моделирования становятся видимы линии тока.

16.4. Течение после препятствий

Обтекание препятствия можно получить, заставив стенки препятствия отражать частицы газа, например, как в разд. 15.2. Можно также ввести источники и стоки. Препятствия, источники и стоки "программируются" *изображением* их на вспомогательных плоскостях битов; они могут быть размещены произвольным образом и иметь произвольные формы. Различная степень трения со стенками программируется в данном случае не введением "коэффициента трения" в определенные уравнения; это достигается изменением текстуры стенки, которую можно сделать имеющей углубления или выступы. Дело каждой отдельной частицы, как ей попасть в эти углубления и выбраться из них; ее импульс будет изменяться в большей или меньшей степени за счет соударений, а трение будет возникать спонтанно как совокупный результат этих блужданий.

Отметим, что граничные условия можно сделать сколь угодно сложными, *нисколько не усложняя программирование эксперимента и не удорожая его проведение*. Более того, поскольку макроскопические величины возникают лишь как средние, взятые по явно представленным микроскопическим конфигурациям, то неустойчивости численных решений и расходящиеся решения исключаются. Когда "действующие лица" являются теми, кем они кажутся, то мы можем спокойно позволить им "делать то, что они должны".

16.5. Другие решеточные газы

На тему решеточных газов можно сыграть много концептуальных и практических вариаций. Здесь мы вкратце представим подход FHP [18], но сначала еще раз взглянем на газ HPP, введенный в гл. 12.

Газ HPP был первоначально определен следующим образом [23]. Рассмотрим ортогональную решетку, состоящую из позиций, соединенных северной, южной, западной и восточной связями. Имеются четыре вида частиц, по одному на каждое направление, и позицию может занять самое большее по одной частице каждого сорта (таким образом, может быть до четырех частиц на место). Обновление осуществляется в цикле из двух шагов. На нулевом шаге каждая частица перемещается вдоль связи от ее текущей позиции к смежной, соответствующей направлению ее движения; на шаге 1 частицы тасуются в каждой позиции способом, аналогичным табл. (12.4). Таким образом, если в этой позиции имеется ровно две частицы, которые прибыли с противоположных направлений, скажем с севера и с юга, то они заменяются парой запад /восток; в противном случае ничего не меняется.

Этот метод можно непосредственно реализовать на CAM. Четыре плоскости битов соответствуют четырем видам частиц, так что используются все четыре бита клетки. На шаге О ("движение") каждая плоскость сдвигается на один шаг в соответствующем направлении, в то время как на шаге 1 ("столкновение") проверяется содержимое четырех центральных битов и выполняются необходимые изменения. За счет использования пользовательской окрестности в CAM можно объединить два шага в один. Это значит, что некоторые сигналы между соседями и фазовые сигналы передаются таблицам внешним образом посредством пользовательского коммутатора, а не изнутри (см. разд. 7.5, 9.7). Один шаг *сдвига* включает выбор одного южного, северного, западного или восточного соседа с каждой плоскости и перемещение его в центральную позицию той же плоскости; но если у нас уже есть эти четыре бита в качестве аргументов справочной таблицы, то мы могли бы также запрограммировать таблицу и для того, чтобы перед их перемещением перетасовать их так, как это предписано шагом *столкновения*. На рис. 16.4 изображено распространение круговой волны в этой более плотной реализации модели HPP. Этот вид схемы является скорее примером разбиения множества состояний каждой клетки (каждый бит состояния клетки используется в качестве соседа в точности одной другой клеткой), чем использования разбиения на блоки.

Нужно отметить, что эта реализация газа HPP использует CAM менее эффективно, чем та, которая применяет окрест-

ность Марголуса; частицы принадлежат двум разделенным пространственно-временным подрешеткам, каждая из которых эволюционирует независимо бдна от другой (см. разд. 14.3). Вместо системы, содержащей *N* частиц, мы приходим к моделированию двух независимых систем, содержащих *N/2* частиц каждая. В случае окрестности Марголуса представлена и моделируется лишь одна подрешетка.

Хотя меньшее число битов, используемых в каждой клетке реализацией окрестности Марголуса, было недостатком когда мы хотели создать изображения с большим числом частиц, видимые в окне размером 256x256 (см. рис. 16.1 и 16.4), оно будет достоинством в следующем разделе, когда мы захотим одновременно запустить две копии одной и той же системы, чтобы сравнить соответствующие позиции на разных плоскостях битов.

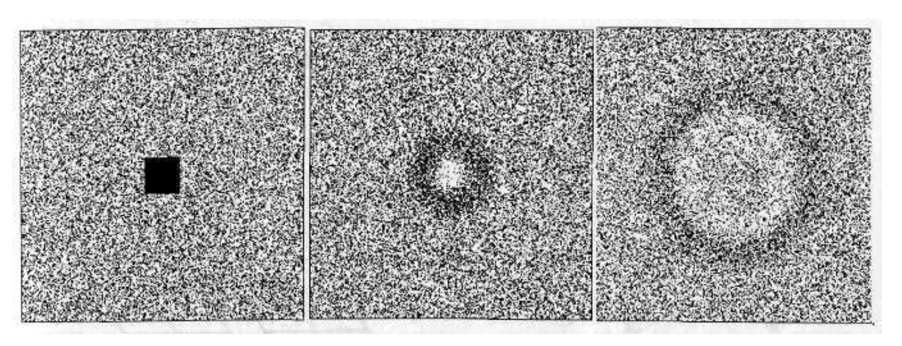


Рис. 16.4. Распространение волн в более плотной реализации модели газа HPP. Для усиления контраста показаны только точки, содержащие три или четыре частицы.

Как мы упомянули ранее, поведение модели HPP отклоняется от решений уравнения Навье-Стокса; даже на макроскопическом уровне вязкость анизотропна - "тень" решетки. Этот недостаток устраняется при помощи модели FHP [18], которая использует шесть видов частиц; т. е. имеется *шесть* направлений перемещения с углом в 60° между двумя смежными направлениями. Решетка гексагональна и каждая позиция может содержать до шести частиц (по одной каждого вида). Как и в моделях HPP и TM, соударения определены так, чтобы сохранялись энергия (число частиц) и импульс.

Простая версия правила FHP приводит к возникновению столкновения всякий раз, когда импульс в какой-либо позиции равен нулю. Если столкновения нет, то все частицы движутся по прямой. В случае же столкновения все частицы в этой по-

зиции отклоняются на 60° по часовой стрелке (или против часовой стрелки) от направления, по которому они следовали.[1]

Модель FHP можно реализовать на CAM, используя окрестность Марголуса на двух плоскостях. Одну ось решетки следует представлять себе отклоненной на 30° от вертикали, как показано на рис. 16.5. Каждый блок представляет одну позицию; из восьми битов блока (по четыре на каждую плоскость) шесть используются для частиц; оставшиеся два являются "запасными" и могут быть использованы, например, чтобы определить препятствия или другие пространственно-зависимые свойства среды (см. разд. 16.7). Правило FHP-GAS, которое мы не будем здесь детально расписывать, использует, как и в приведенной выше модели HPP, цикл в два шага; один шаг используется для перемещения частиц вдоль связей, и один - чтобы выполнить перемещение, вызванное соударением.

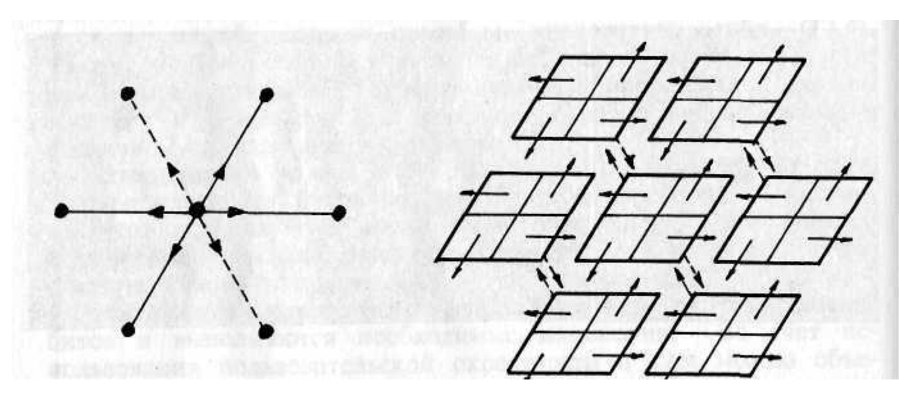


Рис. 16.5. Концептуально блоки окрестности Марголуса расположены шестиугольником. Каждый блок выполняет роль позиции. Сплошные линии относятся к частицам в плоскости 0, штриховые - к частицам в плоскости 1.

Вложение, аналогичное изображенному на рис. 16.5, позволяет реализовать на CAM обычную (не разбитую на части) гексагональную окрестность: достаточно написать правила для окрестности N.MOORE, в которых не используется, скажем, N. EAST и S.WEST (попробуйте применить правила из разд. 6.4 к такой гексагональной окрестности). Применяя пользовательскую окрестность и две машины CAM, можно аналогично реализовать модель FHP с шестью частицами в каждой позиции: использовать отдельные плоскости битов для каждой из шести скоро-

[1] В первоначальном правиле FHP столкновения четырех частиц игнорировались, и эту версию мы использовали для рис. 16.6. На каждом шаге мы использовали одно и то же направление поворота для всех отклонений, и на последовательных шагах направления поворота чередовались.

стей и чередовать шаги, которые перетасовывают частицы в каждой позиции (соударения), с шагами, которые сдвигают плоскости битов (движение). Точно так же, как и в приведенном выше случае газа HPP, эти два шага можно объединить использованием окрестности, в которой каждая клетка присоединена к движущейся на юг плоскости клетки над ней, к движущейся на север плоскости клетки под ней и т. д. В отличие от реализации NYMARG, которую мы описали первой, эта не будет тратить половину времени на шаги, сдвигающие данные. Однако ее было бы труднее использовать для экспериментов с автокорреляцией из следующего раздела.

16.6. Автокорреляции

Функция временной автокорреляции скорости дает ответ на следующий вопрос. Рассмотрим микроскопическую скорость жидкости в некотором месте; насколько *отличной* может стать эта скорость спустя некоторое время t? Аналогичные вопросы могут быть поставлены и для других величин, чем скорость, или для пространственных, а не временных корреляций. Автокорреляции представляют важный (и в эксперименте, и в теории) инструмент для установления связей между макроскопическими и микроскопическими свойствами системы.

Концепция измерения автокорреляции требует рассмотрения двух копий одной и той же системы, сдвинутых по отношению друг к другу на определенное расстояние или время (или и то и другое).[1] Во многих интересных случаях величина корреляции довольно резко падает при увеличении смещения и вскоре тонет в статистическом шуме; чтобы отфильтровать этот шум, требуется большое количество данных. В этой ситуации желательно измерять корреляции в каждой точке массива и среднее по всему массиву; дальнейшее усреднение этих результатов проводится по длительному сеансу моделирования.

Метод измерения разностей, рассмотренный в разд. 9.2, может быть распространен и на изучение автокорреляций. Вместо сравнения в реальном времени историй двух копий системы, начинающихся со слегка отличающихся начальных конфигураций, здесь используется *одна и та же* начальная конфигурация, однако одна копия системы запускается на t шагов позже. (Аналогичный подход используется и для пространственных автокорреляций.)

[1] Если затраты на моделирование чрезмерно велики, может показаться предпочтительнее запустить единственную копию системы, записать ее историю, а затем изучать ее. Как отмечено в разд. 9.2, этот подход может быть очень обременительным.

13-1608

В решеточном газе, таком как HPP-GAS или TM-GAS, каждая клетка блока окрестности Марголуса резервируется для частицы, имеющей определенное направление движения (разд. 12.2, 12.7). Скорость в такой точке может быть определена как 1, если клетка содержит частицу, и как 0, если клетка пустая. Автокорреляция скорости между двумя гомологичными клетками[1] определяется как *произведение* соответствующих скоростей. Таким образом, если одна копия системы прогоняется на плоскости 0, а другая на плоскости 1, то можно так запрограммировать интенсивности на выходе цветовой карты, чтобы он возвращал произведение этих двух плоскостей клетка за клеткой:

```
                        : AUTOCORR-MAP
    ALPHA ALPHA' AND >INTEN       \ Произведение C и C
                  ...             \ Другие цвета по желанию
                        ;         \  для визуальной картинки
```

Здесь мы использовали "AND" вместо "*" (с тем же результатом; см. разд. А. 14), чтобы подчеркнуть, что произведение двух битов имеет однобитовое значение.

Как объяснено в разд. 7.7, сигнал интенсивности с выхода цветовой карты подается на счетчик событий, который, таким образом, будет на каждом шаге интегрировать автокорреляцию по всему массиву. Усреднение этих отсчетов по большому числу шагов - тривиальная задача, оставляемая компьютеру-хозяину.

Теоретические доводы наводят на мысль, что для простого решеточного газа в одном, двух или большем числе измерений функция $v(i)$ временной автокорреляции скорости должна быть *степенью* t (по крайней мере асимптотически, при $t \to \infty$). Предполагается, что показатель этой степени, который на графике функции $v(t)$ в логарифмических координатах является просто наклоном кривой, для больших t в истинно двумерной модели стремится к -Т.

На рис. 16.6 изображены измеренные значения функции $v(t)$ Для трех решеточных газов, упомянутых в этой книге, а именно HPP-GAS, TM-GAS и FHP-GAS.[2] Эти эксперименты, обсуждавшиеся в [36], мы провели в MIT с Жераром Вишняком, используя единственный модуль CAM. Каждая из изображенных точек данных получена накоплением результатов миллиона

[1] То есть данной клеткой и клеткой, соответствующей ей при пространственном или временном сдвиге.

[2] Для нашей реализации FHP-GAS одна копия системы занимает плоскости 0 и 1 (см. рис. 16.5), в то время как вторая копия системы на прогоняется на CAM-B. Чтобы сравнить две соответствующие пары плоскостей, используются два шага; так как обновление/перемещение также требуют двух шагов, то эти сравнения могут быть сделаны без замедления моделирования.

или более сравнений. Весь эксперимент потребовал накопления результатов около 3/4 триллиона сравнений, и время прогона составило около двух с половиной дней.

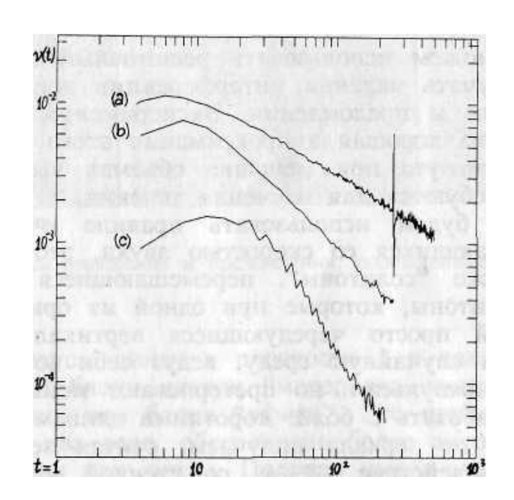


Рис. 16.6. Функция временной корреляции $_u(t)$ для HPP-GAS (а), TM-GAS (о) и FHP-GAS (с).

В экспериментах такого типа широко используются преимущества последовательной по битам, параллельной по плоскостям внутренней архитектуры САМ (см. начало гл. 7 и разд. В.5). На каждом шаге две плоскости битов, содержащие две копии исследуемой системы, последовательно сканируются. Единственная функция AND, аргументами которой служат два потока битов, отслеживает локальные корреляции по ходу моделирования (именно это выполняет слово AUTOCORR-MAP) и передает их счетчику событий (который в этой реализации просто последовательный счетчик), где они накапливаются в течение шага. На этой стадии из потока моделирования, включающего, скажем, сто тысяч битов на шаг, соответствующая информация, состоящая лишь из нескольких битов (содержимого счетчика), уже извлечена, и дальнейшая обработка тривиальна.

16.7. Волновая оптика

Теоретический анализ [23] показывает, что для небольших возмущений равновесия упругие свойства решеточных газов данного вида *линейны.* В этой ситуации распространение возмущения определяется на макроскопическом уровне хорошо знакомым *волновым уравнением*

if.

Поэтому мы можем использовать решеточный газ в качестве "эфира" и изучать явления интерференции волн, их отражения, дифракции и преломления. Эксперименты из разд. 16.1 показывают, что хорошая аппроксимация этого поведения может быть достигнута при меньших объемах вычислений, чем те, которые требуются для изучения течения.

Здесь мы будем использовать правило HPP-GAS. Помимо волн, перемещающихся со скоростью звука, это правило поддерживает также "солитоны", перемещающиеся со скоростью света. Эти солитоны, которые при одной из ориентации представляют собой просто чередующиеся вертикальные полоски, наложенные на случайную среду, ведут себя во многом подобно звуковым импульсам, но претерпевают меньшее рассеяние и позволяют работать с более короткими длинами волн.

Отражение от зеркала получаем, считая зеркало твердой стенкой; взаимодействие HPP-GAS со стенкой можно получить аналогично тому, что делалось для TM-GAS в разд. 15.2. На рис. 16.7 изображено отражение импульса плоской волны сферическим зеркалом.

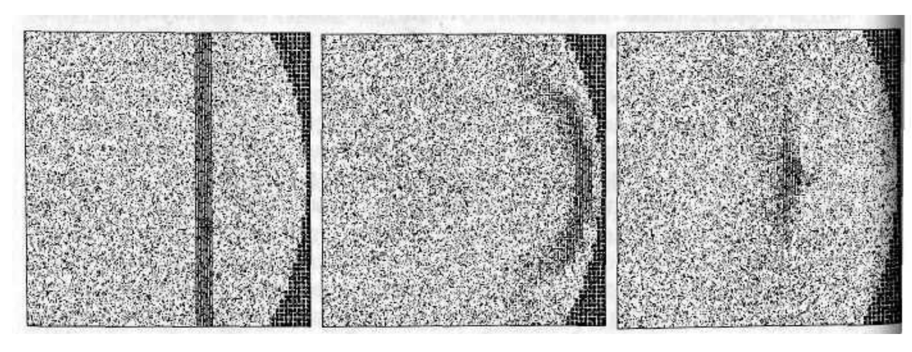


Рис. 16.7. Плоская волна, перемещающаяся по направлению к вогнутому зеркалу (a), показана сразу после отражения (Ь) и сходящейся в фокальную точку (c).

Отражение получить легко. А как насчет преломления? Можем ли мы сделать *линзу?* Для этого нам необходима среда с более высоким коэффициентом преломления, чем "эфир", это значит, что сигналы в этой среде должны перемещаться с уменьшенной скоростью.

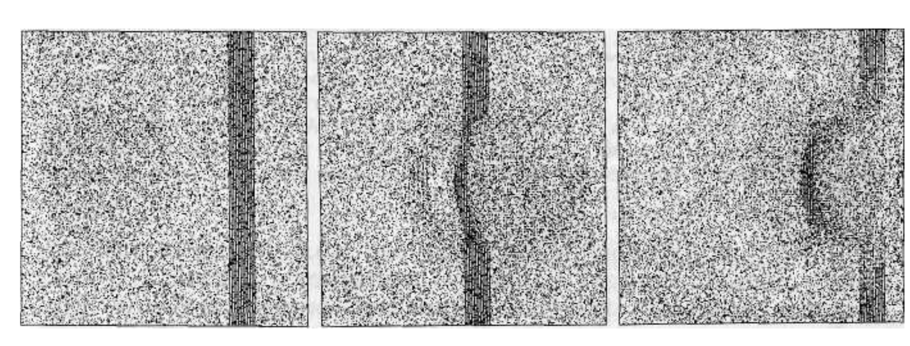


Рис. 16.8. Картины отражения и преломления, порожденные сферической линзой.

Мы нарисуем линзу в плоскости 1 и так модифицируем правило, чтобы блоки клеток "материала линзы" обновлялись вдвое медленнее, чем блоки "эфира". Это приведет к замедлению частиц в два раза, обеспечивая более высокий коэффициент преломления для линзы. Правило имеет следующий вид:

```
N/HARG
                              : LENS (– 0|1)
        CENTER' CW OPP' CCW     \ находится АИ какая-то
                  OR OR OR ;    \    часть линзы внутри
                                \    блока?

                              : REFRACT
           LENS PHASE AND IF    \ если внутри линзы,
                CENTER ELSE     \    отметить время при PHASE=1
              COLLISION IF      \ иначе вести себя
                  CW ELSE       \    как обычный
                 OPP THEN       \    HPP-GAS
                      THEN
                     >PLNO
               CENTER >PLN1 ;   \ линза остается
                                \ неизменной
                              : ООП-CYCLE
          0 IS PHASE
          0 IS <0R6-HV> STEP    \ Два шага с PHASE=O :
          3 IS <0R6-HV> STEP    \    все действует
          1 IS <PHASE>
          0 IS <ORG-HV> STEP    \ Два шага с PHASE=1:
          3 IS <ORG-HV> STEP ;  \    действует только эфир
MAKE-TABLE REFRACT
MAKE-CYCLE OOH-CYCLE
```

Напомним, что HPP-GAS выполняет один шаг на четной решетке и один на нечетной. Мы хотим замедлить активность внутри линзы, но без разделения двух элементов пары шагов (это попарное объединение шагов является существенной чертой правила). Внутри линзы за парой активных шагов будет следовать пара "холостых", и т. д. поочередно. Слово PHASE используется для различения активных и холостых шагов. Все остальное так же, как и в обычном правиле HPP-GAS; определение COLLISION было дано в 12.4 и здесь не повторяется.

Этот эксперимент показан на рис. 16.8. Часть волны, которая первой попадает на линзу, замедляется первой; это искривляет волновой фронт и заставляет волну сходиться. Поскольку мы используем круглую линзу, то сходящиеся лучи обнаружат *сферическую аберрацию:* вместо резко выделенной фокальной точки они дадут *каустику* - картинку, которая наблюдается, когда свет отражается внутри чашки, наполненной молоком. Заметим, что имеется также слабая отраженная волна. Это не артефакт, порожденный моделированием; в *обратимой* среде теория предсказывает отражения всякий раз, когда имеется резкий разрыв коэффициента преломления, т. е. скорости распространения информации. *Если информация поступает в какую-то точку быстрее, чем может пройти дальше, то некоторая часть ее должна отразиться, поскольку она не может быть потеряна.*

Глава 17

КОЛЛЕКТИВНЫЕ ЯВЛЕНИЯ

Каждый отдельный человек старается по возможности употребить свой капитал так, чтобы полученный продукт обладал наибольшей стоимостью. Обычно он не имеет в виду содействовать общественной пользе и не сознает, насколько он содействует ей. Он имеет в виду лишь собственный интерес, лишь собственную выгоду. Причем в этом случае он невидимой рукой направляется к цели, которая совсем и не входила в его намерения. Преследуя свои собственные интересы, он часто более эффективно содействует интересам общества, чем тогда, когда сознательно стремится делать то.

[Адам Смит]

В природе часто бывает так, что много похожих элементов (например, молекул, животных) взаимодействует между собой таким образом, что возникают широкомасштабные феномены поведения, не присущие отдельным элементам. В качестве примера можно указать обсуждавшиеся уже газы: отдельные молекулы, подчиняющиеся довольно простым законам столкновения, в совокупности демонстрируют коллективное поведение, которое может быть весьма разнообразным и сложным - звуковые волны, вихри, турбулентность.

С точки зрения моделирования интересен тот факт, что в сущности то же самое макроскопическое поведение можно получить, если начать с "индивидуумов", которые намного проще реальных молекул, а именно с клеток клеточного автомата. Только некоторые свойства (в нашем случае - сохранение частиц и импульса) оказываются способны проявляться на все более крупных масштабах, вплоть до макроскопического уровня; остальные свойства становятся неважными на достаточно больших масштабах.

В этой главе мы покажем, как клеточный автомат можно использовать для моделирования других коллективных явлений, таких как намагничивание, и для изучения фазовых переходов, происходящих при достижении некоторыми параметрами критических значений. Эта феноменология первоначально наблюдалась в физике, но аналогичные эффекты были описаны и в других областях (биологии, экономике), где взаимодейст-

вующие индивидуумы могут быть гораздо сложнее молекул газа. Как и в случае с газами, существенные моменты этих коллективных явлений можно, как правило, воспроизвести клеточными автоматами, поскольку они обладают важной особенностью - локальностью взаимодействия большого скопления сходных индивидуумов.

17.1, Критические параметры и фазовые переходы

В идеальном газе состояние равновесия, соответствующее данным глобальным ограничениям (полная энергия, импульс), уникально и недифференцировано, т. е. (а) все начальные состояния, подчиняющиеся данным ограничениям, в конечном счете эволюционируют к одному и тому же макроскопическому состоянию; (Ь) в отсутствие внешних полей выборочные части системы, отобранные из разных мест и в разное время, неразличимы.

Вкратце, равновесие идеального газа малоинтересно. Верно ли это для равновесия в общем случае?

Интуитивно картина такова. Многие свойства вещества - это результат борьбы сил притяжения, стремящихся создать упорядоченные локальные структуры, и теплоты, которая стремится разрушить эту упорядоченность. Ряд характерных явлений возникает вблизи так называемой *критической температуры,* когда стремление к достижению большей локальной упорядоченности и разрушающее влияние теплоты уравновешивают друг друга, и нет преобладания того или другого.[1]

Феноменология коллективных явлений чрезвычайно разнообразна. Другие параметры могут взаимодействовать с температурой, устанавливая критические наборы значений, вблизи которых плавное изменение некоторых переменных ведет к весьма резкому (и интересному - хотя бы только по этой причине) изменению структуры.

17.2. Системы Изинга

В качестве общей концептуальной парадигмы систем, обладающих более богатой феноменологией равновесия, чем идеальные газы, рассмотрим системы *Изинга* - класс моделей, первоначально введенных для изучения магнитных материалов. Мы сохраним некоторые вызывающие содержательные ассоциации термины и понятия, пришедшие из физического контекста.

[1] В классическом "идеальном газе" нет сил притяжения: беспорядок распределяется по возможности равномерно, критическая температура отсутствует.

Спины. "Индивидуумы" системы Изинга - это *спины* (которые можно упрощенно представлять в виде маленьких магнитов), организованные в упорядоченный массив. В отличие от частиц газа спины занимают *фиксированные положения;* единственное, что может меняться - это *ориентация* спина в пространстве. Мы в дальнейшем ограничимся случаем, когда возможны только две ориентации, условно обозначенные "вверх" (Т) и "вниз" (|). (Можно представлять себе ось "вверх/вниз" перпендикулярной плоскости массива). Для одномерного массива конфигурация системы может быть следующей:

• • •ТТШШШШИ---

Два спина назовем *параллельными,* если они указывают в одном направлении (ТТ)> и *антипараллельными,* если они указывают в противоположных направлениях (jj).

Знакомство с обыкновенными магнитами наводит на мысль, что спины воздействуют друг на друга посредством сил, зависящих от их взаимной ориентации, расстояния между ними и природы окружающей их среды. В случае систем Изинга аналогия сохраняется; однако детали спаривания спинов (которое в обычной физической ситуации включает квантовомеханические явления) будут установлены на основе весьма абстрактных соображений, а взаимодействовать будут только спины, расположенные *непосредственно рядом* друг с другом (никаких сил дальнодействия).

Энергия спаривания. Рассмотрим самый простой случай системы спинов, в которой силы, действующие между соседними спинами, стремятся повернуть их в одном направлении *(выровнять).* Связь между двумя спинами можно представить в виде пружины, которая при параллельных спинах находится в нормальном состоянии и растянута, когда спины антипараллельны. Каждая растянутая пружина несет одну единицу энергии. В конфигурации (17.1), где энергия каждой пары обозначена для ясности 0 или 1,

a b o d e

ТЦо!---, (17.1)

спин *а* выровнен с обоими своими соседями. Если попытаться повернуть его из положения | в положение i , чтобы получить конфигурацию

a Ь c d e /17 *У\

нужно противодействовать двум пружинам одновременно: одной справа, и одной слева, и при этом система *приобретет* две единицы энергии. Аналогично, поворачивая спин *c* в (17.1), можно *высвободить* две единицы энергии. С другой стороны, спин *Ъ* в (17.1) находится в *безразличном* энергетическом положении: если его повернуть, то пружина справа растянется, а слева вернется в нормальное состояние; общая энергия системы не изменится.

Преобразования, сохраняющие энергию. В физике энергия эволюционирующей с течением времени изолированной системы сохраняется. Мы останемся верными этому принципу и для моделей Изинга, представленных ниже, независимо от того, насколько решительные упрощения будут введены в других отношениях.

В реальных спиновых системах может быть много других "мест", в которых запасается энергия, кроме межспиновых связей, и существует множество моделей, подобных моделям Изинга, в которых учитывается это обстоятельство. Некоторые из них мы изучим позднее. Сейчас же рассмотрим простую модель, в которой важна только энергия межспиновых связей. Сможем ли мы написать правило клеточного автомата, которое заставит систему развиваться нетривиальным образом, подчиняясь в то же время политике строгого сохранения энергии?

Предположим, что вы обновляете по одному спину за один шаг - скажем, выбирая его произвольно; могут быть два возможных исхода такой модификации: спин "перевернут" и спин "не перевернут". Если вы перевернули спин, то будут модифицированы только те связи, которые непосредственно его окружают. Тогда принцип сохранения энергии позволит вам перевернуть только те спины, которые находятся в *безразличном* энергетическом состоянии, как, например, спин *Ь* в конфигурации

$$\begin{array}{c} \quad\ \ \boldsymbol{a}\ \ \boldsymbol{b}\ \ \boldsymbol{c}\ \ \boldsymbol{d}\ \ \boldsymbol{e} \\ \cdots\uparrow 0\uparrow 0\uparrow 1\downarrow 0\downarrow 1\uparrow 1\downarrow\cdots. \end{array}$$

Как только *Ь* окажется перевернут, спин *a,* который до этого был "заморожен" энергетическими ограничениями, станет хорошим кандидатом на перевертывание:

$$\begin{array}{c} \quad\ \ \boldsymbol{a}\ \ \boldsymbol{b}\ \ \boldsymbol{c}\ \ \boldsymbol{d}\ \ \boldsymbol{e} \\ \cdots\uparrow 0\uparrow 1\downarrow 0\downarrow 0\downarrow 1\uparrow 1\downarrow\cdots. \end{array}$$

Таким образом, одно изменение влечет за собой другое, и с течением времени такая система может заметно измениться.

Теперь предположим, что вы хотите заставить систему изменяться настолько быстро, насколько это возможно. Прежде всего каждый выбранный вами спин, который по энергетическим соображениям можно перевернуть, вы всегда переворачиваете. Далее, возможно, вам понадобится помощник: вы выбираете себе спин и приступаете к работе с ним, в то время как помощник работает с другим спином. Само собой разумеется, что вы оба переворачиваете только энергетически безразличные спины.

До тех пор пока вы сохраняете некоторое минимальное расстояние между вами, все идет хорошо; однако предположим, что вы наметили заняться спином c в (17.3), а ваш помощник - спином d, стоящим рядом с с. В обоих местах энергетическая картина выглядит подходящей для перевертывания, и каждый из вас переворачивает свой спин. Но, взглянув на итоговую картину

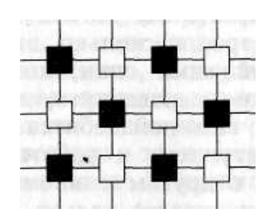
$$\begin{array}{ccccccccccccc} & & a & & b & & c & & d & & e & & \\ \cdots & \uparrow & 0 & \uparrow & 0 & \uparrow & 0 & \uparrow & 0 & \uparrow & 1 & \downarrow & \cdots \end{array} \qquad (17.5)$$

вы обнаружите, что общая энергия изменилась с 3 до 1! Спин энергетически безразличен только в предположении, что когда вы перевернете его, соседние спины сохранят свою ориентацию. (Это аналогично ситуации "освоителей целины" из разд. 9.5.) Если хочется (а) применять *локальное* правило, (Ь) *одновременно* обновлять максимально возможное число спинов и (с) *сохранять энергию,* то безопасная стратегия состоит в следующем: сначала за один шаг обновляем все четно пронумерованные спины, на следующем шаге - все нечетно пронумерованные и так далее, чередуя.

Аналогичные рассуждения применимы и для двумерного случая. Теперь спины организованы в виде прямоугольной решетки и соединены со своими четырьмя соседями связями с энергией 0 или 1, как на следующей диаграмме:

$$\begin{array}{ccccccc} \boxed{\downarrow} & 1 & \boxed{\uparrow} & 1 & \boxed{\downarrow} & 0 & \boxed{\downarrow} \\ 0 & & 0 & & 0 & & 1 \\ \boxed{\downarrow} & 1 & \boxed{\uparrow} & 1 & \boxed{\downarrow} & 0 & \boxed{\uparrow} \\ 0 & & 1 & & 1 & & 1 \\ \boxed{\downarrow} & 0 & \boxed{\downarrow} & 1 & \boxed{\uparrow} & 1 & \boxed{\downarrow} \end{array}, \qquad (17.6)$$

Для обновления решетку разбивают на две подрешетки точно так же, как на обычной шахматной доске,



(17.7)

и эти подрешетки обновляют поочередно. Заметим, что каждая связь соединяет две клетки разного цвета, поэтому на каждом шаге ее энергия может быть изменена только "на одном конце"; в этом случае вполне достаточно чисто локальной схемы учета, чтобы быть уверенным, что энергия сохраняется.

Для дальнейшего будет более удобным обозначить "вверх" через 1 и "вниз" через 0.

В следующих разделах мы будем пользоваться структурами спинов типа структуры Изинга при моделировании

- изолированной системы, энергия которой строго сохраняется;

- системы, которая обменивается энергией со своим окружением либо

  - *явным образом,* запоминая индивидуальные взаимодействия с тем, чтобы сумма энергий системы и окружения сохранялась, либо

  - *неявным образом,* считая окружение массивным тепловым резервуаром или тепловой ванной, который с некоторой вероятностью отдает или принимает энергию отсистемы, но сам по себе не испытывает существенного влияния со стороны этих обменов. В последнем случае энергия сохраняется в статистическом смысле.

Традиционным считается неявный (тепловая ванна) подход, который не один десяток раз был использован на моделях Изинга в теоретических и практических работах. Явный подход был введен и интенсивно исследован Крютцем [13]. Подход, основанный на изолированной системе (см. следующий раздел), начал применяться в широкомасштабном моделировании лишь совсем недавно (Херрманн [26]). Он основан на правиле Q2R, открытом Вишняком [66] с использованием САМ, и применен на практике Помо [45] для создания консервативной динамики в модели Изинга.

17.3. Только спины

Описанная выше система Изинга может быть реализована с помощью CAM следующим образом. Посредством пространственных фаз реализуется структура шахматной доски (см. разд. 11.6) с тем, чтобы четно и нечетно нумерованные клетки можно было поочередно модифицировать. Спины располагаются в плоскости 0; состояние 1 означает спин "вверх", о - спин "вниз". В активной в данный момент подрешетке любой спин, который можно перекинуть без приобретения или потери энергии (при четырех соседях это соответствует наличию в точности двух соседей разного типа), *будет* перевернут. Правило, называемое SPINS-ONLY, работает в контексте следующего рабочего цикла:

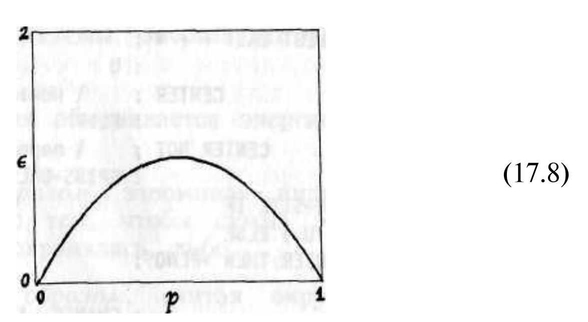
```
NEW-EXPERIMENT
N/VONN 8./HV
                                        : ACTIVE-SITE  (-- F/T)
                         8.HORZ &VERT = ;
                                        : 4SUM  ( -- 0,..,4 )
           NORTH SOUTH WEST EAST + + + ;
                                        : U
                                CENTER ;     \ неизменяемый
                                        : FLIP
                            CENTER NOT ;     \ переворачивается
                                        : SPINS-ONLY
                   ACTIVE-SITE IF
        4SUM { U U FLIP U U } ELSE
                      CENTER THEN >PLNO ;
MAKE-TABLE SPINS-ONLY
                                        : CHANGE-LATTICE
                  <0RG-H> NOT IS <ORG-H> ;
                                        : ALT-LATTICE
                   STEP  CHANGE-LATTICE ;
MAKE-CYCLE ALT-LATTICE
```

Слово ACTIVE-SITE выявляет те спины, которые принадлежат активной в данный момент шахматной подрешетке (&HORZ=&VERT); рабочий цикл ALT-LATTICE осуществляет чередование активных подрешеток и дополняет "фазирование" &HORZ на кажом шаге. 4SUM подсчитывает количество соседей со спином "вверх"; SPINS-ONLY использует это число при решении вопроса о перевороте спина; спин будет перевернут, если он находится в энергетически безразличном состоянии, т. е. если он окружен в точности двумя парами соседей каждого типа (4SUM=2).

Давайте произвольным образом заполним плоскость спинами, половина которых - "вверх", половина - "вниз", и запустим правило: как и можно было ожидать, порядка не прибавилось, ничего существенного не произошло.

Теперь возьмем 1/4 спинов "вверх" - нечто вроде жидкого "случайного супа". Вскоре этот суп примет вид "эмульсии", распавшись на небольшие черные и белые островки, разделенные размытыми неустойчивыми границами (рис. 17.1а). Спустя некоторое время количество белого и черного станет примерно одинаковым; если вы шаг за шагом измеряете и записываете долю *и* спинов "вверх" (это, как показано в разд. 7.7, можно сделать, не прерывая моделирования), то вы обнаружите, что эта доля остается близкой к 1/2 с малыми короткоживущими флуктуациями.

Заметим, что число единиц определенно *не* сохраняется. С другой стороны, общая длина границы между черными и белыми областями остается неизменной; действительно, каждый сегмент границы единичной длины разделяет два смежных спина с разной ориентацией и, таким образом, представляет единицу энергии связи. Для произвольной начальной конфигурации, в которой вероятность спина "вверх" равна *p,* энергия *e* (ожидаемая), приходящаяся на одну клетку\ дается зависимостью $e = 4p(1-p)$, график которой приведен ниже:



(17.8)

Эта зависимость может быть использована для калибровки в терминах энергии генератора случайных чисел, используемого для получения начальной конфигурации.[2]

Динамика SPINS-ONLY обратима и в действительности эквивалентна (см. разд. 14.3) обратимому правилу второго порядка [66]. Если вы остановите процесс моделирования, выполните команду CHANGE-LATTICE и продолжите моделирование дальше, то система будет развиваться в обратном по времени направле-

[1] Здесь удобно разделить полную энергию массива *E* на число клеток, чтобы полученный диапазон значений энергии не зависел от размера массива.

[2] Наибольшая энергия, которую можно получить при случайном распределении спинов, - это $e = 1$; она достигается при р = 1. Однако максимально возможное значение есть $e = 2$, и его можно получить при вполне порядоченной конфигурации, в которой одна подрешетка содержит все единицы, а другая - все нули; в этом случае все связи возбужденные.

нии. Если система развивалась вперед в течение двух часов, то при "движении" в обратном направлении доля *и* спинов "вверх" будет большую часть этих двух часов колебаться около значения 1/2 (что не несет в себе какого-либо особого смысла), а за последние несколько секунд резко упадет до 1/4, т.е. до своего начального значения.

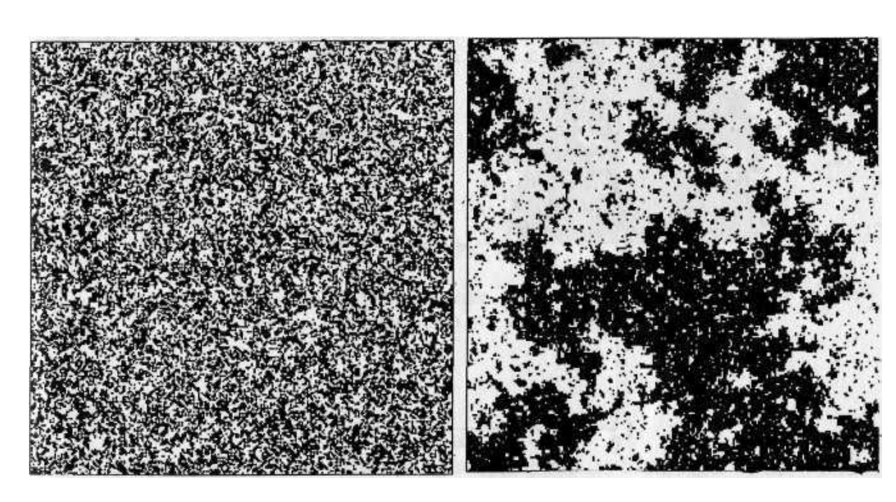


Рис. 17.1. Равновесная конфигурация (а) при энергии выше критической и (Ь) при критической энергии.

В литературе вместо доли *и* спинов "вверх" принято говорить о *намагниченности* образца, определяемой формулой /х = u - (1 - u) (т.е. из доли спинов "вверх" вычитают долю спинов "вниз"). Когда все спины "вверх", ^ =-\-\ , а когда "вниз", *ц* = -1 .

По мере того как начальная произвольная доля спинов "вверх" и, следовательно, энергия *е* постепенно уменьшается, средний размер черных и белых островков становится больше, при этом продолжает иметь место равновесное распределение черного и белого цветов (*ц* = 0). Это продолжается до тех пор, пока энергия *е* не достигнет своего критического значения $e_{крит}$ (рис. 17.1Ь); в этот момент *ц* очень внезапно начинает уходить от нулевого значения. Поскольку энергия продолжает уменьшаться и за критическое значение, то *р,* примет одно из двух приведенных на рис. 17.2 значений. Оба различных положения равновесия равновероятны: одно с преобладанием белого, другое - черного. Кривая была получена Чарльзом Беннеттом экспериментально с помощью САМ.

Когда *e* все еще немного меньше $e_{крт}$ и, следовательно, два значения /*и* чуть-чуть отличаются друг от друга, то иногда можно обнаружить, что система совершенно неожиданно перескакивает из одного состояния равновесия в другое. При низких энергиях такое спонтанное обращение намагниченности становится невозможным.

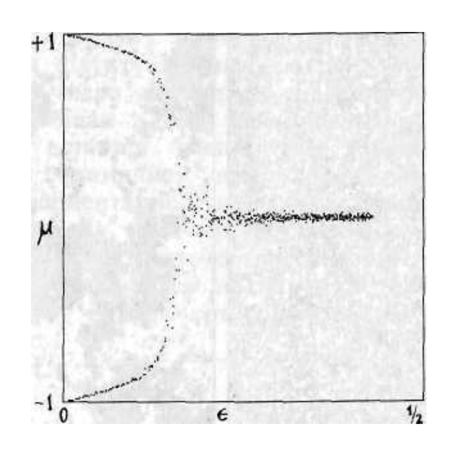


Рис. 17.2. Намагниченность *И* и энергия *e*. Ниже критического значения $e_{крит}$ возможны два значения для

По этой причине, если нужно наблюдать нарушение симметрии, о возможности которого говорит рис. 17.2, то необходимо уделить внимание подготовке симметричных начальных условий.[1] Одна из процедур - это подготовка неравновесного состояния, для которого $e < e_{крит}$, но м=1/2. Например, можно заполнить половину массива, скажем, 5% единиц, а вторую половину - 5% нулей. Такой баланс неустойчив: граница между этими двумя областями будет дико "плясать" (рис. 17.3Ь), сами области - распадаться и в конце концов один цвет будет преобладать над другим (с равными шансами для обоих).

Размытость критической точки зависит от размера массива. Только предельный случай бесконечного массива делает эту точку вполне локализованной.

[1] Заметим, что в (17.8) одно и то же значение *e* можно получить для двух различных значений *p*. Ниже критической точки, однако, выбор более низкого значения *p* приведет с очень высокой вероятностью к более низкому значению и (и аналогично при выборе более высокого значения р).

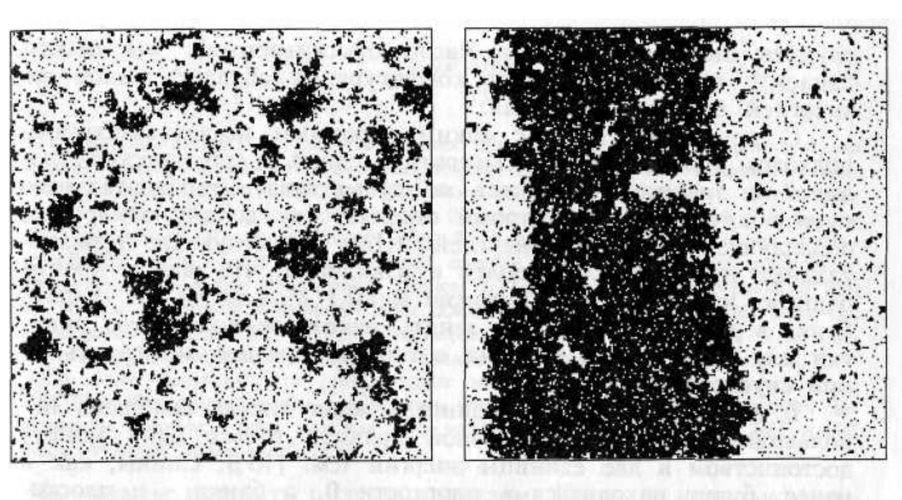


Рис. 17.3. (а) Одно из двух равновесных состояний с энергией ниже критической; (Ь) Приближение к равновесию из неравновесного состояния.

17.4. Банки энергии

Для данной системы спинов рассмотрим множество Г(£) всех конфигураций, обладающих данной энергией *E*. Как свойства данного множества меняются в зависимости от *E* - один из наиболее интересных вопросов статистической механики.

Даже для систем с умеренным числом спинов *N* размер этого множества огромен (число конфигураций растет как экспонента от *N*); ясно, что невозможно создать и исследовать все конфигурации с данной энергией. Однако относительно небольшое подмножество Г может дать хорошее приближение статистических свойств всего множества, если процедура выборки подмножества в некотором смысле "справедлива".

Для создания такого подмножества можно было бы произвольно выбирать новые конфигурации, подсчитывать их энергию и оставлять только те, энергия которых равна *E*. Такой подход чрезвычайно малоэффективен и едва ли когда-либо применялся на практике. Динамику, рассмотренную в предыдущем разделе, можно считать более экономичным подходом к достижению той же цели. Дана конфигурация с известной энергией *E*. Сериями локальных изменений создадим новые конфигурации с той же энергией - тем самым осуществляя более или менее случайное блуждание по множеству *Г(E)*. В этом случае нет необходимости явным образом вычислять энергию конфигурации (только понимать, что в большинстве

случаев она не входит в число отыскиваемых): мы просто "знаем", что каждая новая конфигурация обладает такой же энергией *E,* что и исходная.

С другой стороны, с таким скромным исследовательским подходом можно навсегда застрять в одной энергетической "долине" с энергией *E:* может не существовать горизонтального пути вокруг барьера, который отделяет эту долину от другой, обладающей такой же энергией. (Или существующие горизонтальные пути настолько долги и извилисты, что шансы пройти один из них до конца ничтожно малы.) Можем ли мы создать банк, в котором могли бы занять некоторое количество энергии для преодоления подъема в гору, а позднее возвратить ее при спуске с горы?

В следующей модели Изинга каждая клетка снабжена небольшой "копилкой", способной хранить только одну монету достоинством в две единицы энергии (см. [13]). Спины, как и ранее, будут находиться в плоскости 0, а банки - в плоскости 1. Правила для каждой клетки следующие:

- Если вы можете перевернуть спин без приобретения или потери энергии - сделайте это.
- Если перевертывание спина могло бы освободить две единицы энергии и банк пуст, то переверните спин и поместите энергию в банк.
- Если перевертывание потребует двух единиц энергии, а банк полон - возьмите энергию из банка и переверните спин.

Новое правило SPIN-BANK, которое определено ниже, использует то же окружение рабочего цикла, что и SPINS-ONLY предыдущего примера.[1]

```
                                  : BONDS (-- 0,..,4}
             NORTH SOUTH WEST EAST + + +
                CENTER IF 4 SWAP - THEN ;
                                  : GET
                       CENTER' { 0 3 } ;
                                  : PUT
                       CENTER' { 3 0 } ;
                                  : SPINS-BANK
                             CENTERS
                      ACTIVE-SITE  IF
              BONDS { 0 GET 1 PUT 0 }
```

[1] Плоскость 1 теперь используется для банков, а не, как обычно, для получения ECHO от плоскости 0 (см. разд. 3.2), и цвета на экране будут отражать эту роль. Если желательно использовать ECHO, ТО ПОД него можно отвести одну плоскость CAM-B или выходную функцию, чтобы получить тот же результат, не расходуя на это целую плоскость.

```
XOR THEN
   >PLNA ;
```

С помощью **CENTERS** мы заносим в стек объединенное состояние спина и его банка. Чтобы решить, как изменить это состояние, мы далее используем значение энергии связи **BONDS**, взяв операцию XOR ЭТОГО значения с соответствующей маской. Значение маски, равное 0, означает, что изменений нет, равное 1 - дополнение только спинового бита, 3 - дополнение как спина, так и банка. **BONDS** - это просто сумма четырех соседей при равном 0 бите спина и ее дополнение до 4 при бите спина, равном 1. Когда энергия связи равна 0 или 4, то мы ничего не изменяем. Если B0NDS=2, то мы перевертываем спин без обращения к банку. **GET** и **PUT,** связанные со значением энергии 1 и 3 соответственно, проверяют пригодность банка для желаемой операции; если "да" (т.е. банк пригоден), то они создают маску, которая дополняет оба бита **CENTERS** (бит спина, чтобы перевернуть спин; бит банка, чтобы поместить две единицы энергии в банк или взять их из банка). В других случаях **GET** и PUT выдают маску "изменений нет".

Мы можем поиграть с этим правилом. Пусть в начальном состоянии банки пусты. Заполним плоскость спинов, скажем, на 50% единицами. Прогоним модель в течение нескольких секунд, пока спины и банки не окажутся в равновесии. Поскольку теперь некоторая часть начальной энергии связи занесена в банки, то конфигурация в спиновой плоскости очень сильно будет напоминать ту, что представлена на рис. 17.1a, хотя там все начиналось с меньшим запасом энергии. Теперь очистим плоскость 1, стерев тем самым все сбережения в банках; конфигурация спинов изменится, приспосабливаясь к такой ситуации. При этом вкладов в банки будет производиться больше - пока две формы наличности (связи и сбережения) не придут в равновесие друг с другом (рис. 17.4a). Вы можете повторять эту процедуру, приводя систему в состояния со все более низкими значениями энергии.

В промежутках, когда мы не вмешиваемся в систему, ее полная энергия остается строго постоянной. Однако при равновесии энергия спиновой компоненты системы постоянна лишь приблизительно; повторные измерения обнаруживают разброс энергии около ее центрального значения *E*. Поэтому энергия уже не может служить параметром для описания положения равновесия. Таким параметром является *температура*[1]: кража энергии из банков "охлаждает" систему спинов, добавление энергии - "нагревает" ее. Как и в предыдущей модели, фазовый переход, аналогичный переходу рис. 17.2, будет иметь место при определенном критическом значении (энергии системы спин /банк или температуры ее спинового компонента).

[1] В данном контексте понятие "температура" корректно определено лишь в пределе *бесконечной* спиновой системы; этот термин приобретет строгий смысл даже для *конечной* спиновой системы в рамках модели, обсуждаемой в следующем разделе.

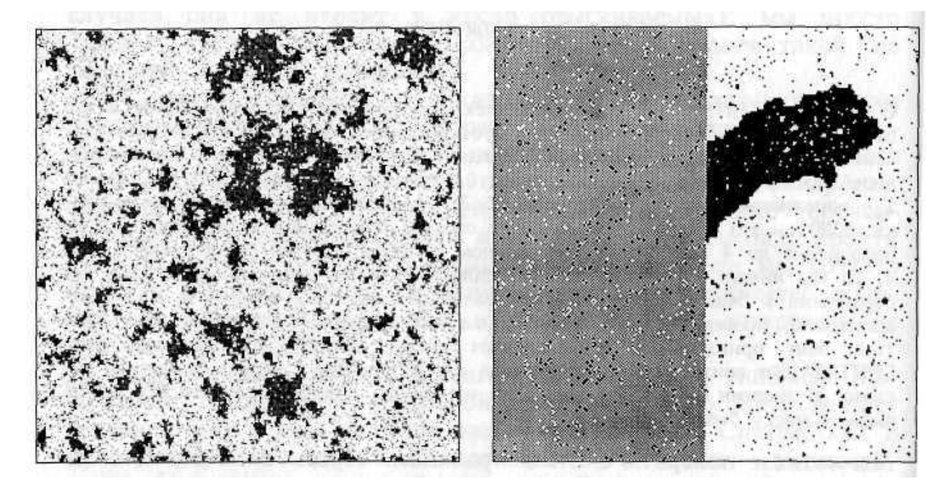


Рис. 17.4. (а) Состояние равновесия модели Изинга с банками энергии при температуре чуть ниже критической точки. Спины "вверх" окрашены темно-серым (банки пусты) или черным (банки заполнены), а спины "вниз" окрашены соответственно белым и светло-серым. (Ь) Фазы шахматного типа появляются из-за роста температуры выше верхней критической точки; правая половина массива показана промаскированной образцом типа шахматной доски.

Как мы видели, при низкой температуре система спинов точно разделяется на две *фазы* (состояния вещества, обладающие хорошо различимыми составом и микроскопической структурой); в одной фазе большинство спинов направлено вверх, в другой - вниз. Области одной фазы отделены от областей другой границами, которые становятся более четкими и более прямолинейными при понижении температуры (см. разд. 5.4). Что произойдет, если мы будем накачивать в банки все больше и больше энергии? Представленный на рис. I7.4b результат можно было бы угадать, исходя из подсказки, содержащейся в примечании 2 к стр. 206. Система вновь разделится на две фазы; однако теперь соседние спины *антипараллельны* в каждой фазе и, таким образом, микроскопическая структура - это структура типа шахматной доски. В одной из фаз единицы расположены на четной подрешетке, в другой - на нечетной. (Рисунок *11АЪ* - это картина, наблюдаемая на разделенном на две части экране. В левой части видны только спины, организованные в домены. Последние обладают шахматным порядком и отличаются только относительной фазой. Правая часть картины - это правая часть той же конфигурации, но уже так скоррелированной посредством маски шахматного формата данной фазы, чтобы четко показать две разновидно-

сти доменов.) В такой системе спинов увеличение энергии сверх некоторого значения вызывает скорее порядок, чем беспорядок.[1] График зависимости от температуры доли v, занимаемой одной из фаз шахматного порядка, некоторое время идет горизонтально на уровне 1/2 при росте температуры, но в конце концов разделяется на две кривые при дальнейшей подкачке энергии (по аналогии с рис. 17.2). Таким образом, наряду с критической температурой, связанной с параллельными фазами, система обладает второй критической температурой, связанной с антипараллельными фазами.

17.5. Тепловая ванна

В предыдущем примере (спины с банками) невольно возникало искушение рассматривать энергию связей как *потенциальную энергию,* а энергию, запасенную в каждом банке, - как *кинетическую энергию,* связанную с соответствующим спином: если спин обладает достаточной кинетической энергией, он может использовать ее для преодоления потенциального барьера некоторой высоты. Однако такая аналогия сама по себе не приводит к полезным обобщениям; на самом же деле имеет значение то, что мы ввели новый энергетический "текущий счет" в дополнение к текущему счету энергии межспиновых связей и то, что эти два текущих счета взаимосвязаны так, что они могут достигать статистического равновесия друг с другом.

Емкость банков может быть увеличена[2], соседним банкам может быть позволена передача энергии друг другу. Такие сделки могут регулироваться более сложными правилами (например, спину может быть запрещено совершать подряд две банковские\ сделки). При дальнейшем движении в этом направлении механизм банковского комплекса с точки зрения отдельного спина станет слишком сложен для прослеживания в деталях, и (как и в реальной жизни) сделки с банком станут предприниматься из вероятностных соображений. Концепция

[1] Это явление хорошо известно в физике и привело к тому, что появилось выражение "отрицательная температура". Ее можно представлять себе так: ось температуры свернута в кольцо, и с ростом температуры мы в конце концов снова приближаемся к нулю, проходя через отрицательные температуры с убывающими абсолютными величинами. С учетом такой интерпретации обе упорядоченные текстуры (целиком параллельная и антипараллельная) спиновой решетки соответствуют температурам, близким к нулю: одна отрицательной, а другая положительной.

[2] Располагая, например, одним модулем САМ, довольно легко использовать биты третьей плоскости, чтобы банки могли манипулировать с монетами стоимостью как в четыре, так и в две энергетические единицы (таким образом, изолированные нули и единицы также получают возможность переворачиваться) .

теплоты была придумана для работы с энергией, которая проявляется в такой стохастической форме.

С макроскопической точки зрения массив спинов можно представить себе выложенным на *тепловом субстрате,* удельная теплоемкость которого пропорциональна размеру банков, а удельная теплопроводность пропорциональна легкости суммарного обмена между банками. Вблизи равновесия вклады и изъятия вкладов, сделанные индивидуальными спинами за достаточно продолжительное время прогона усредняются к нулевому итогу, и у банков есть время уравненить свои капиталы. В этом случае весь комплекс банков можно рассматривать как единый тепловой резервуар, способность которого к удовлетворению потребности в энергии описывается единственным параметром - *температурой.*

С практической точки зрения тепловой подход избавляет нас от многих переменных, представляющих банки, и заменяет их генератором случайных чисел, который с некоторой вероятностью принимает решение о каждой банковской сделке.

Элементарный результат равновесной статистической механики состоит в том, что температура тепловой ванны *Т* точно определена тогда и только тогда, когда для любого изменения энергии *АЕ* вероятность *Р(АЕ)* удовлетворения запроса на заем энергии Д.Е и вероятностьР(-Д£)приема вклада того же количества энергии связаны соотношением

-» •

где *к* - коэффициент пропорциональности, определяющий единицы измерения температуры.

Для проведения эксперимента на САМ, соответствующего данной ситуации, отметим, что в системе спинов, которую мы рассматриваем, перевертывание спина может приводить лишь к следующим значениям изменения энергии системы *АЕ:*

-4, -2, 0, +2, +4

(что поддается простой проверке). Таким образом, соответствующие вероятности

$$P_{-4}, P_{-2}, P_0, P_2, P_4$$

должны быть установлены так, что

(^у и ?

Р-4 Р-2 Р-2

где величина энергии, описывающая связь между спинами, представлена как /, чтобы сделать соображения размерности более понятными. Поскольку мы имеем дело только с отношениями вероятностей, то у нас есть некоторая свобода выбора самих вероятностей; для простоты выберем $p_0 = p_2 = p^* - 1$, так что окончательные значения имеют вид

$$P^2, p, 1, 1, 1, \qquad (17.10)$$

при этом

$$T = \frac{13}{\text{fclogp}'} \qquad (17.11)$$

Последнее отношение позволяет нам прокалибровать вероятность *p,* выдаваемую генератором случайных чисел, в терминах температуры Г.

Как обычно, поместим спины в плоскость 0 и воспользуемся CAM-B как генератором случайных чисел. В CAM-B каждая плоскость независимо выдает 1 с вероятностью *p,* следовательно, логически перемножая два бита (операция "AND"), получают 1 с вероятностью p^2. Доступ к CAM-в требует, чтобы вспомогательное назначение для CAM-A было &/CENTERS, что делает для нас невозможным доступ к пространственным фазам. Чтобы избежать применения пользовательской окрестности, мы синтезируем псевдососеда ACTIVE-SITE, инициализируя плоскость 1 паттерном типа шахматной доски и выполняя на каждом шаге операцию дополнения этой плоскости, почти так, как мы это делали в разд. 10.1. Правило для этой системы выглядит следующим образом:

```
CAM-A N/VONN «./CENTERS
                                              ACTIVE-SITE (-- 0/1)
                                     CENTER'      \ шахм. доска
                                              CHANGE-LATTICE
                          CENTER "NOT >PLN1       \ дплн. UIXMT
MAKE-TABLE CHANGE-LATTICE
                                              P ( -- 0/1)
                                     &CENTER
                                              P2 ( -- 0/1)
                       &CENTER &CENTER' AND
                                              4SUM
               NORTH SOUTH WEST EAST + + +
                                              DELTA ( -- 0....4)
                                  CENTER IF
                                  4SUM ELSE
```

```
                    4 4SUM - THEN ;
                                  : SPIN-CANON
                              CENTER
                    ACTIVE-SITE IF
          DELTA { P2 P 1 1 1 }
                            XOR THEN
                              >PLNO ;
MAKE-TABLE SPIN-CANON
```

По состоянию клетки и ее соседей DELTA вычисляет значение обмена энергией в форме, пригодной для использования оператором выбора (т. е. Д.Е/2+2, а не *&E)*. SPIN-CANON использует эту величину для выбора одного из пяти значений вероятности, установленных в (17.10). Биты Р и Р2, связанные с вероятностями *p* и p^2, получают в САМ-В от генератора случайных чисел.

Системы, подобные модели SPIN-CANON, которые свободно обмениваются энергией с внешним тепловым резервуаром, называют *каноническими*. В противовес этому замкнутую детерминированную систему, подобную модели SPINS-ONLY разд. 17.3, называют *микроканонической*. Тем не менее, эти модели ведут себя почти одинаково, так как в микроканонической модели каждая часть системы обменивается энергией с "тепловым резервуаром", который состоит из остальной части системы. На рис. 17.5 представлены: типичная конфигурация (а) и снимок системы с длительной выдержкой, полученный при температуре чуть ниже критической (Ь). Заметим, что в отличие от рис. 17.3 а здесь нет изолированных спинов, ориентация которых длительное время остается фиксированной: тепло[1] вой резервуар способен отдавать или принимать с соответствующей вероятностью *любое* количество энергии.[1]

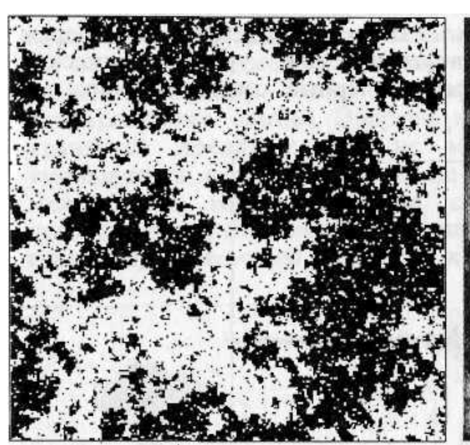
На рис. 17.6 мы показываем кривую зависимости намагниченности от температуры для нашей системы; эта кривая была экспериментально получена Чарльзом Беннеттом на большом числе опытов с САМ.[2]

С практической точки зрения введение тепловой ванны оправдано более быстрым и более полным достижением равновесного состояния. С другой стороны, явный подход - банки и "только спины" - сохраняет некоторые черты *динамики* реальной физической системы, которые отсутствуют в подходе "тепловая ванна". "Поскольку температура системы обусловлена ее

[1] Банкиры с капиталом в два бита, способные одолжить всю необходимую энергию для переворота изолированного спина, позволяют системе преодолевать энергетические барьеры почти так же эффективно, как тепловой резервуар.

[2] С целью минимизировать корреляции, порожденные генератором случайных чисел, и тем самым получить результаты, непосредственно сравнимые с имеющимися в литературе, в этом эксперименте применялся метод, упомянутый в конце разд. 15.6.

внутренним строением, то тепловой поток и удельная теплопроводность могут быть исследованы численно. Не ясно, имеют ли эти идеи какой-либо смысл в обычном моделировании методом Монте-Карло" [13].



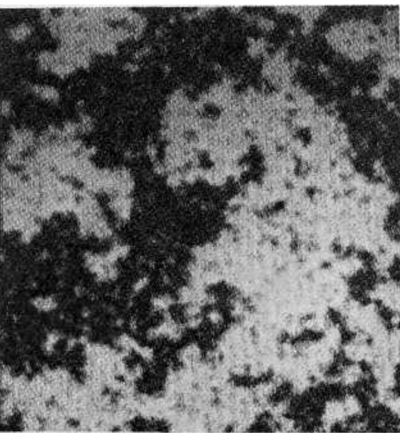


Рис. 17.5. (а) Типичная конфигурация спинов в канонической модели вблизи критической температуры. (Ь) Временная экспозиция при тех же условиях; никакие спины не сохраняют неограниченное время своей ориентации.

17.6. Отображение энергии

До сих пор в модели Изинга энергия играла роль весьма абстрактной величины - *отношения* между объектами (спинами), не являясь сама по себе *объектом.*[1] Прежде чем строить модели, в которых энергия рассматривается как самостоятельная переменная состояния, полезно придать энергии более ощутимое существование, непосредственно отобразив ее на экране.

Вернемся к модели "только спины" разд. 17.3, в которой присутствует только один вид энергии - энергия межспиновых связей. Эта энергия, так сказать, сидит на границах между смежными клетками, которых в два раза больше чем клеток, и поэтому мы не можем отвести целый пиксель каждому элементу энергии. Проще всего связать с каждым пикселем полную энергию связи соответствующего спина и представить пять возможных значений (о, 1,..., 4) пятью различными цветами или пятью оттенками серого цвета. Заметим, что каждая

[1] В решеточных газах, рассмотренных ранее, энергия является чисто кинетической и связана с частицами взаимно однозначным соответствием, так что это отличие несущественно.

связь вносит свой вклад в два смежных пикселя, поскольку она "принадлежит" двум спинам и, следовательно, будет "размытой" на картинке. Мы сможем получить более четкую картинку, если свяжем с каждым пикселем только две связи, на север и на запад от него, а значит, каждая связь появится только в одном месте экрана. Так мы и сделаем и выберем для представления трех энергетических уровней пикселя (О, 1, 2) три цвета: белый, серый и черный соответственно.

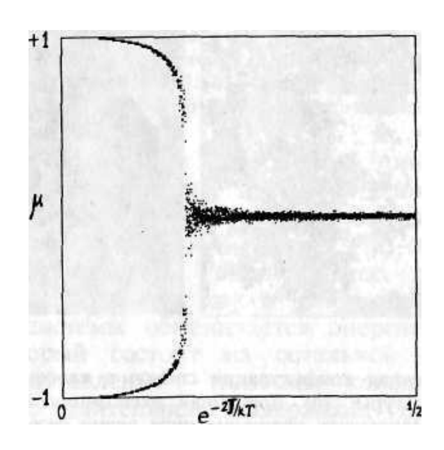


Рис. 17.6. Намагниченность /л модели канонического ансамбля в зависимости от распределения вероятности метода Монте-Карло. Обратите внимание на резкий переход при температуре $T_{к\ ит}$.

Естественно, мы хотели бы отображать энергию *в процессе моделирования,* с тем чтобы непосредственно наблюдать ее эволюцию; поэтому нам нужны две справочные таблицы: одна для вычисления следующего состояния клетки и его засылки в плоскости битов *(функция перехода),* другая - для вычисления энергии связи клетки и посылки ее на экран монитора *(выходная функция).* Каждая из этих таблиц должна целиком видеть окрестность. Как указывалось в разд. 7.7, вычисление выходной функции - одно из предназначений вспомогательных таблиц.

Энергию северной и западной связей вычисляют с помощью вспомогательной таблицы, запрограммированной следующим образом:

```
                         : ENERGY-DISPLAY
          NORTH CENTER XOR >AUX0      \ северная связь
           WEST CENTER XOR >AUX1  ;   \ западная связь
MAKE-TABLE ENERGY-DISPLAY
```

Команда SNOWFUNCTION предписывает цветовой карте брать свои входные данные из этой таблицы, а не сразу из плоскостей битов.

Если мы проведем эксперимент рис. 17.3 по нашей схеме, то мы увидим "канаты" энергии, извивающиеся по экрану, как на рис. 17.7; большое количество энергии сконцентрировано на границах между доменами со спинами "вверх" и доменами со спинами "вниз", остальная часть энергии окружает изолированные спины. Энергия не только сохраняется, но сохраняется *локально:* она не может увеличиваться здесь и уменьшаться в другом месте, не проходя при этом через промежуточные точки. Другими словами, энергия подчиняется *уравнению неразрывности,* и на экране ее можно наблюдать гладко *протекающей* от точки к точке.

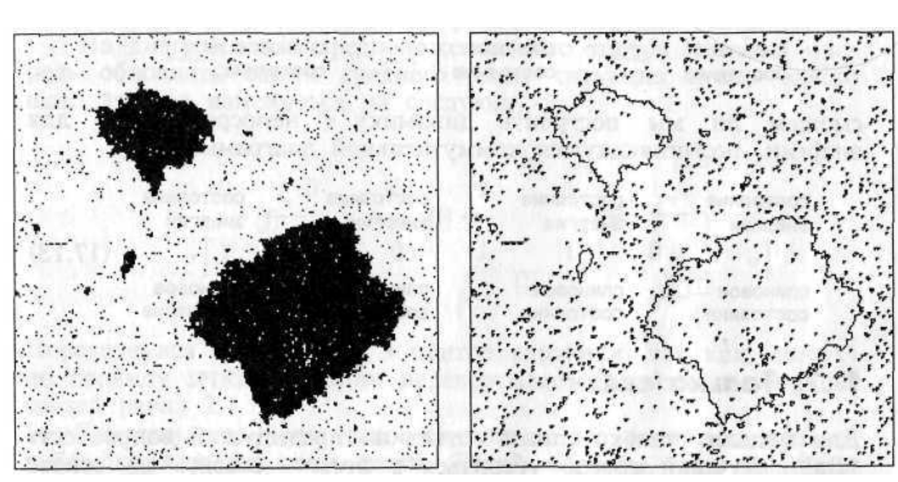


Рис. **17.7.** (а) Типичная конфигурация спинов; (Ь) та же конфигурация, но отображающая энергию, а не спины.

Предположим на время, что мы, вместо того чтобы увязывать их по две на пиксель, отображаем на экране каждую связь отдельно.[1] Картинка энергии при этом все еще содержит меньше информации, чем плоскость спинов, т. е., зная состояние спинов, всегда можно вычислить энергию связей, но, зная энергию связей, остаешься в неопределенности относительно состояния спинов. Например, наблюдая на экране резкую линию энергии, вы знаете, что спины по разные стороны от нее ориентированы в противоположных направлениях, но вы не в

[1] Мы могли бы покрасить северную связь в красный цвет, западную - в зеленый; тогда желтый цвет означал бы наличие обеих связей.

силах сказать, с *какой* именно стороны от границы расположены спины, ориентированные, скажем, "вверх".

В то же время движения, наблюдаемые в энергетическом представлении на экране, кажется, имеют свою собственную определенную логику. Вопрос, который мы собираемся задать, состоит в следующем: "Исходя из данной энергетической конфигурации, обладаем ли мы достаточной информацией для определения *последующей энергетической конфигурации?* Сможем ли мы представить динамику одной только энергии?".

Более формально, если *т* - динамика спинов и и - функция энергии, как показано на диаграмме

состояние энергии ↑υ спиновое состояние →τ состояние энергии ↑υ спиновое состояние →τ состояние энергии ↑υ спиновое состояние →τ (17.12)

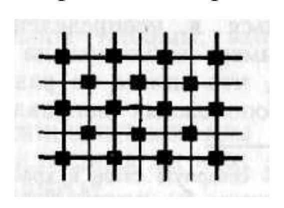
сможем ли мы построить динамику *т'* непосредственно для энергии, подчиняющуюся коммутативной диаграмме

состояние энергии	состояние энергии	состояние энергии	состояние энергии	
				(17.13)
спиновое состояние	спиновое состояние	спиновое состояние	спиновое состояние	

17.7. Только связи

Для модели "только спины" ответ на предыдущий вопрос есть "Да!". Лучший способ убедиться в этом - создать на основе клеточного автомата модель для системы спинов, в которой именно энергии связей, а не спины, играют рать переменных состояния.

В такой модели содержимое каждой клетки соответствует состоянию связи. На диаграмме квадраты изображают клетки, как и на рис. 12.1, а спины (которые не представлены в модели в явном виде) нужно воображать сидящими на пересечении двух жирных или двух тонких линий в позициях, помеченных маленькими черными квадратиками.



Следующая диаграмма показывает типичную конфигурацию значений спинов и значений связей в этом представлении:

Ш m ш Ш
0 0 i 0 1 i
Ш Ш Ш
1 0 1 0 1 0
Ш Ш Ш ш

Заметим, что по отношению к предыдущей модели (см. диаграмму 17.6), ось решетки спинов повернута на угол 45°, а масштаб увеличен (умножен на >/2). *Площадь,* соответствующая данной части системы, теперь вдвое больше (что неудивительно, поскольку связей в два раза больше, чем спинов).

На диаграмме (в предположении, что теперь подошла очередь обновлять спины среднего ряда) ситуация после одного шага должна измениться на следующую:

Ш m Ш Ш
0 0 0 1 1 1
Ш Ш Ш
1 0 0 1 1 0
Ш m Ш Ш

Перевернулся только спин в центре картинки, так как, согласно правилу SPINS-ONLY, спин перевертывается, когда энергия его связей равна 2.

Заметим, что когда такое случается, это влияет на все четыре окружаюшие спин связи. Таким образом, для эволюции связей (нам потребуется правило для блока, состоящего из четырех клеток. Более того, обновление спинов на жирных линиях чередуют их обновлением на тонких линиях (так как SPINS-ONLY использует чередующиеся подрешетки типа шахматной доски); следовательно, чередующееся разбиение на блоки для' окрестности Марголуса полностью нас устроит.

Представляя единицу энергии в виде "частицы", мы получим правило обновления связей, которое соответствует правилу обновления спинов SPINS-ONLY:



(17.14)

все другие элементы правила суть "без изменений". Это правило, которое мы назовем BONDS-ONLY, приведено на фото 14.

За исключением масштабного коэффициента, структура того же типа, что и в 17.7Ь.

Сравнивая эти две модели, мы сможем получить некоторое понимание динамики системы Изинга.

Прежде всего, если энергетическая картинка не полна в том смысле, что в ней недостает некоторой информации, то как она может столь достоверно отслеживать развитие картинки спинов? Сколько информации здесь действительно теряется?

Пусть дана спиновая модель; перевернем все спины. Энергия при этом не изменится, но и эволюция не изменится: единственное различие состоит в том, что черные области станут белыми, а белые - черными; эта динамика инвариантна по отношению к операции дополнения[1]. Решение интерпретировать состояние клетки либо как "спин вверх", либо как "спин вниз" есть выбор одной из двух возможностей; если решение принято, то состояния остальных клеток полностью определяются значениями связей. Бит, связанный с таким выбором, - вот единственный элемент информации, который утерян энергетической картинкой. Значение этого бита *не изменяется во времени:* так стоит ли заботиться о том, чтобы протащить его через весь процесс моделирования? Именно этого энергетической модели удается избежать.

Итак, динамика системы спинов может быть разложена на две компоненты. Одна состоит из единственного бита, представляющего собой константу; другая неотличима от динамики системы энергий связи.

Осознав это обстоятельство, мы можем задать следующие вопросы: "Если новая модель ухитряется избежать бремени "мертвого" бита, то как же она приходит к тому, чтобы использовать в два раза больше памяти по сравнению со старой моделью? Для чего нужна вся эта дополнительная информация?".

Рассмотрим конфигурацию связей схемы (17.15а) и попытаемся воссоздать подходящую конфигурацию для соседних спинов:

(a) [□] 0 0 [□] [□] 0 1 [□]   (b) [↓] 0 0 [↓] [↓] 0 1 [?]   (17.15)

[1] В физическом контрагенте системы Изинга такая симметрия была бы разрушена наличием внешнего магнитного поля. Взаимодействие спинов при наличии поля требует, безусловно, более сложной модели.

Если мы выберем для верхнего спина диаграммы (17.15b), то два смежных с ним спина обязаны иметь такую же ориентацию, поскольку соответствующие связи обладают нулевой энергией; однако ориентация последнего спина (помеченного [Тр не может быть установлена согласующимся с другими двумя связями способом: одна из них предполагает ориентацию Щ, другая Ш.

Итак, многие возможные конфигурации связей несовместимы ни с какой конфигурацией спинов[1], и в этом смысле они бесполезны в качестве начальных для энергетической модели: при создании начальной конфигурации таких моделей для десяти любых произвольно выбранных связей найдется десять, которые выбраны вынужденно.

Ввиду такой избыточности, обладает ли энергетическая модель какими-либо компенсирующими эту избыточность свойствами? Сможем ли мы найти для этой избыточности хорошее применение?

17.8. Спиновые стекла

Вернемся к разд. 17.2, где мы рассмотрели ряд спинов, соединенных упругими связями. Если два соседних спина параллельны, то связь между ними "ненапряженная"; если они антипараллельны - "возбужденная". Связь такого типа называют *ферромагнитной.*

В различных физических явлениях процесс спаривания спинов выгодней описать с помощью *антиферромагнитных* связей, которые обладают противоположными свойствами, т. е. не напряжены, когда спины *антипараллельны.* Конечно, поведение антиферромагнитных спиновых систем отличается от поведения (ферромагнитных систем. Однако в отсутствие внешнего магнитного поля это различие тривиально: если каждый второй спин ферромагнитной системы "читать" так, как будто его ориентация противоположна, то динамика данной системы идентична динамике антиферромагнитной системы. При таком условии одна модель может интерпретироваться двояко (см. симметрию между параллельными фазами при низкой темпера-

[1] Чтобы понять, какие конфигурации запрещены, рассмотрим два (из четырех возможных) разбиения на блоки 2x2, которые окрестность Марголуса не использует. Четность энергии любого из этих блоков (т. е. является ли значение энергии четным или нечетным) остается неизменной при перевертывании одного спина - или, по этой причине, при перевертывании произвольного числа спинов. Если энергетическая конфигурация состоит из одних нулей (что заведомо разрешено, ибо это соответствует ситуации, когда спины параллельны), то все блоки четные; поэтому разрешенные энергетические конфигурации - те, у которых все блоки имеют четную четность.

туре и антипараллельными фазами при высокой, о чем упомянуто в конце разд. 17.4).

Существуют также системы, называемые *спиновыми стеклами,* которые лучше всего моделировать, предположив, что случайно выбранная часть связей ферромагнитные, а остальные антиферромагнитные. Таким образом, в модели спиновых стекол природа каждой связи должна быть строго определена, как это показано на диаграмме (17.16), где "=" обозначает ферромагнитную связь, а "*ф*" - антиферромагнитную.

$$\begin{array}{ccccccc}
\boxed{\downarrow} & 1 = & \boxed{\uparrow} & 0 \neq & \boxed{\downarrow} & 1 \neq & \boxed{\downarrow} \\
0 = & & 0 = & & 1 \neq & & 1 = \\
\boxed{\downarrow} & 1 = & \boxed{\uparrow} & 1 = & \boxed{\downarrow} & 0 = & \boxed{\uparrow} \\
0 = & & 0 \neq & & 1 = & & 1 = \\
\boxed{\downarrow} & 1 \neq & \boxed{\downarrow} & 1 = & \boxed{\uparrow} & 1 = & \boxed{\downarrow}
\end{array} , \qquad \mathbf{(17.16)}$$

Теперь энергия связи зависит не только от ориентации связываемых ею спинов, но и от типа (ферромагнитная, антиферромагнитная) самой связи.

Спиновые стекла представляют собой важную концептуальную модель для изучения порядка и беспорядка в веществе [41], а также для некоторых новых подходов к оптимизации [39, 40]. В наиболее простых физических системах состояние равновесия при низкой температуре, или *основное состояние,* уникально. Некоторые системы Изинга, как мы могли заметить в предыдущих разделах, проявляют два различных основных состояния, и этого достаточно, чтобы сделать такие системы заслуживающими внимания. Спиновые стекла проявляют *множество* основных состояний; это их свойство позволяет нетривиальным образом хранить и обрабатывать информацию.

Поскольку общее число связей в два раза превосходит число спинов, то, если мы хотим промоделировать систему спиновых стекол на клеточном автомате, в котором переменные состояния соответствуют *спинам,* как в разд. 17.3, нам понадобятся *три* бита информации на каждую клетку, т. е. состояние спина и, например, тип северной и западной связей (см. начало разд. 17.6). Информацию о связях можно запомнить в двух вспомогательных плоскостях битов[1]; заметим, что эти плоскости битов не нужно обновлять, поскольку тип каждой отдельной связи - это фиксированный параметр, а не переменная состояния. Мы не станем утомлять читателя деталями реализации, за исключением одного замечания: две решет-

[1] В САМ можно было бы использовать для этих целей плоскости 2 и 3 и обычную окрестность, чтобы позволить САМ-А увидеть больше, чем только &CENTERS.

ки (одна для спинов, другая для связей) слегка отличаются по структуре и сдвинуты по отношению к друг другу; с точки зрения реализации это незначительное неудобство.

Когда мы обратимся к модели клеточного автомата, в которой клетки представляют состояния связей, а не состояния спинов, как было в предыдущем разделе для систем Изинга, то нас ждет приятный сюрприз. Простое правило BONDSONLY (17.14), которое мы повторим здесь,

(17.17)

вполне пригодно для моделирования новой системы, и избыточность, которую мы там отметили (дополнительное бремя информации, которое приходилось нести в процессе моделирования) вполне достаточна, чтобы позволить нам представить действие связей двух разных типов. Проблема (17.15а) - поиск распределения спинов, совместимого с данным распределением связей, - здесь всегда имеет решение, так как неизвестными являются не только спины, но и типы связей; вот одно из возможных решений:

(a) 0 0 0 1 (b) 0= 0= 0= 1≠ (17.18)

В действительности же существует множество решений, но все они приводят к одной и той же динамике для энергии. Как и ранее, энергетическая модель отбрасывает некоторую информацию, которая не имеет отношения к динамике; однако в этом случае объем отброшенной информации больше одного бита: по одному биту на каждую клетку. Сбросив это бремя, энергетическая модель ухитряется отразить динамику системы спиновых стекол, используя только два бита на клетку (а не три, как в спиновой модели)[1].

[1] Преобразование, которое может быть применено к системе без воздействия на ее динамику, называют калибровочным преобразованием; здесь одна "энергетическая" система моделирует целый класс "спиновых" систем, которые эквивалентны с точностью до калибровочного преобразования.

Обозначение (17.17) ясно показывает, что в этой модели энергию рассматривают как неразрушимую материальную частицу, которая может перемещаться с некоторыми ограничениями; сохранение энергии и обратимость сделаны очевидными.

В заключение следует сказать, что некоторая разновидность систем может привлечь наше внимание в силу своей теоретической значимости, и возникает искушение считать вопросы о том, как конкретно ее моделировать, не слишком существенными. Однако тщательное изучение методов реализации и возможных здесь вариантов может привести не только к более компактным и эффективным моделям, но и к моделям, позволяющим лучше понять саму систему.

Глава 18

ВЫЧИСЛЕНИЯ НА ОСНОВЕ БАЛЛИСТИЧЕСКОГО МЕТОДА

Какого типа строительные блоки необходимо иметь в наличии, чтобы построить компьютер?

Соображения, развитые в последние несколько десятилетий математической логикой и информатикой, показывают, что если проблема только в возможности, а не в скорости и эффективности, то очень простая аппаратура, если ее достаточно много, способна выполнять самые сложные вычислительные задачи, которые вообще могут быть выполнены *любыми* аппаратными средствами (см. разд. *5.5).* В принципе развитие, жизнь, интеллект могут иметь место в любом мире, управляемом очень простым правилом клеточного автомата.

Упомянутые выше соображения не затрагивают вопроса о том, *обратимы ли* примитивные механизмы, используемые при конструировании; на самом деле современные компьютеры построены на базе *неинвертируемых* логических элементов. Например, схема логического "И", определенная таблицей вход/выход

p	q	p AND q
0	0	0
0	1	0
1	0	0
1	1	1

вводит в вычисления необратимый шаг: когда выходом является 0, вы не можете с уверенностью сказать, каков был вход; если принять, что все четыре входные комбинации равновероятны, то операция AND (И) стирает примерно 1.19 бит информации.

Однако насколько нам известно, микроскопические явления в физике строго обратимы; как же тогда люди ухитряются строить и эксплуатировать компьютеры, содержащие необратимые логические элементы, подобные AND-вентилю? На самом деле происходит следующее. Необратимое поведение логического элемента *моделируется* довольно большим и сложным (по микроскопическим меркам) устройством, функционирующим с

помощью *обратимых* механизмов. Оказывается, что та информация, которую стирает логический элемент, не уничтожается совсем (физически это невозможно) - она просто превращается в тепло, которое отводится системой охлаждения; одновременно подача новых сигналов осуществляется источником питания. Всего этого не избежать, пока мы настаиваем на построении компьютеров на необратимой логике (см. содержательную работу Ландауэра [32]).

Возможно ли создать компьютер на основе *обратимых* логических элементов [4]? Возможна ли реализация таких элементов на уровне микроскопической физики?[1]

Представленная здесь модель вычислений основана на обратимых механизмах такого рода, которые рассматриваются в классической механике (последняя в действительности еще не является настоящей физикой, а только ее полезной идеализацией). Мы также реализуем эту модель в виде клеточного автомата. При этом мы докажем, что (а) с помощью клеточного автомата можно легко моделировать некоторые физические явления и (Ь) даже когда наложены ограничения микроскопической обратимости, эти модели достаточно мощны, чтобы проявлять сколь угодно сложное поведение.

18.1. Модель вычислений посредством биллиардных шаров

В ходе исследований, связанных с наиболее глубинными физическими основами вычислений, Эдвард Фредкин из MIT придумал модель цифровых вычислений [17], которая явным образом отражает некоторые основные свойства физики, в частности обратимость микроскопических процессов.

В этой двумерной модели одинаковые шары конечного диаметра движутся с постоянной скоростью и упруго сталкиваются друг с другом и плоскими зеркалами. Вычисление программируется начальными условиями системы и выполняется обычной динамикой столкновений. Бит информации представлен наличием или отсутствием шара в данном месте в данное время; провода представлены возможными траекториями шаров, при необходимости направляемых зеркалами; логические операции выполняются там, где могут столкнуться два шара (наличие или отсутствие шара на данной траектории может повлиять через столкновение на то, будет или нет присутствовать шар на другой траектории).

[1] Даже если не рассматривать обратимости, то могут быть трудности другой природы.

Модель вычислений посредством "биллиардных шаров" основана на идеализированном описании газа, которое в сущности идентично модели, исторически взятой физиками за основу кинетической теории. Газ представляют себе как множество сфер конечного диаметра, которые претерпевают упругие столкновения друг с другом и со стенками сосуда; механика столкновений обусловлена близкодействующими отталкивающими силами. Новизна модели биллиардных шаров состоит в направлении внимания на детальное развитие во времени отдельного микроскопического состояния, а не на какие-то макроскопические величины, определенные на статистическом распределении состояний.

Такая кинетическая модель есть классическая механическая система, и она подчиняется непрерывной динамике - положения в пространстве, времена, скорости и массы являются вещественными переменными. Чтобы заставить систему выполнить цифровые вычисления, мы можем воспользоваться тем фактом, что целые числа - это частный случай вещественных чисел: подходящим образом ограничивая начальные условия системы, мы можем заставить непрерывную динамику совершать цифровую обработку. Более конкретно, мы (а) установим для шаров (которые соответствуют молекулам газа) специфические начальные условия, (Ь) придадим множеству зеркал (которые соответствуют стенкам сосуда) весьма специфическое пространственное расположение, (с) будем смотреть на систему только в дискретные (с одинаковыми интервалами между ними) моменты времени. В результате получим обратимую механическую систему, способную осуществлять произвольные вычисления.

Заинтересованному читателю можем порекомендовать [17, 35], где дано подробное изложение этой темы. Следующие краткие пояснения будут достаточны для наших целей.

(

Декартова сетка. Начальное положение каждого шара - узел двумерной декартовой сетки; движение осуществляется "вдоль" линий сетки в одном из четырех направлений. Все шары движутся с одинаковой скоростью из одного узла в ближайший к нему за единицу времени. Шаг сетки выбирается таким, чтобы шары сталкивались в узлах. Все столкновения происходят под прямым углом, так что по истечении одного временного шага после столкновения шары по-прежнему находятся в узлах сетки. Неподвижные зеркала установлены так, чтобы шары ударяли в них, находясь в узлах и оставаясь, таким образом, в узлах сетки.

Шары в качестве сигналов. Наличие или отсутствие шара в любом узле сетки можно рассматривать как двоичную переменную, связанную с этим узлом и принимающую значение 1 или 0 (шар в узле или шара в узле нет соответственно) в целочисленные моменты времени. Корреляции между такими переменными отражают движение самих шаров. В частности, можно говорить о двоичных "сигналах", движущихся в пространстве и взаимодействующих друг с другом.

Столкновения как вентили. В модели биллиардных шаров *каждое место, где может произойти столкновение, можно представлять себе как булевский логический вентиль.* На рис. 18.1 пусть *p, q* означают наличие или отсутствие в данный момент шаров с указанными положением и направлением движения. Переменные *p* и *q* будут рассматриваться как входные сигналы для вентиля, находящегося в точке пересечения двух траекторий; точно так же переменные, связанные с четырьмя указанными точками на выходящих траекториях в последующий момент времени (на рисунке - после четырех временных шагов), будут представлять выходные сигналы. Ясно, что эти выходные сигналы примут в указанном на рисунке порядке значения *pq, pq, pq* и снова *pq*. Другими словами, если есть шары на *обоих* входах, то эти шары столкнутся и проследуют внешними выходными траекториями; если только один шар на входе, то этот шар проследует по прямой и выйдет по внутренней выходной траектории. Ясно, что если на входе нет шаров, то нет их и на выходе.

Поскольку вентиль взаимодействия рис. 18.1 может реализовать функцию AND, а при условии что один из входов сохраняют постоянным (постоянный поток шаров), и функцию NOT, то этот вентиль - универсальный логический элемент.

Зеркала как направляющие. Для того чтобы из вентилей создать схему, необходимо установить соответствующие взаимосвязи, т. е. в нужные моменты времени направлять шары от одного места столкновения к другому. В частности, поскольку мы рассматриваем двумерную систему, то необходимо обеспечить способ осуществлять пересечение сигналов. Всем этим требованиям можно удовлетворить введением зеркал. Как показано на рис. 18.2, заставляя шары сталкиваться с неподвижными зеркалами, можно легко отклонить в сторону траекторию шара, сдвинуть ее вбок, ввести произвольную задержку и обеспечить корректное пересечение (из рис. 18.2d видно, что *сигналы* пересекаются, хотя шары этого не делают). Конечно, не нужно никаких особых мер для *тривиальных* пересечений, когда логика или временная диаграмма таковы, что два шара одновременно не могут присутствовать в точке пересечения.

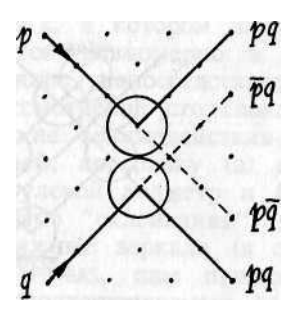


Рис. 18.1. "Вентиль взаимодействия" - способ реализации логических функций на основе столкновений шаров.

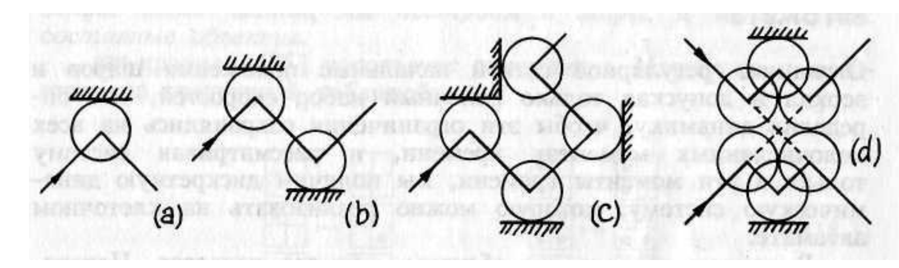


Рис. 18.2. Зеркала (заштрихованы) могут быть использованы для: (а) отклонения под прямым углом траектории шара; (Ь) сдвига траектории; (с) введения задержки; (d) реализации нетривиального пересечения.

В заключение скажем, что на основе описанных выше механизмов можно синтезировать логические элементы и желаемым образом их соединять. Поскольку двоичный сигнал кодируют одиночным шаром и требуется небольшая пауза между потоками шаров, чтобы их нужным способом направлять, то вычисления можно организовать по конвейерной схеме, чтобы все этапы вычислений осуществлялись одновременно.

При заданных ограничениях на начальные условия группа шаров, находящихся на одной вертикальной линии и имеющих одинаковую горизонтальную скорость, сохранит эту выстроенность по вертикали до тех пор, пока они будут сталкиваться только друг с другом и с горизонтальными зеркалами. Это позволяет использовать простые графические методы, чтобы получить подходящие геометрические схемы и временные диаграммы для сложных схем столкновений (см. рис. 18.3) и упростить процедуру поддержания общей синхронизации, необходимой для эффективной реализации конвейерной схемы.

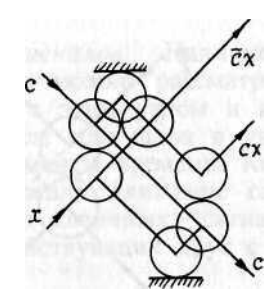


Рис. 18.3. Простая реализация "переключательного вентиля" (двусторонний демультиплексор).

18.2. Обратимый компьютер на основе клеточного автомата

Ограничив регулярной сеткой начальные положения шаров и зеркал и допуская только конечный набор скоростей, так определив динамику, чтобы эти ограничения сохранялись на всех целочисленных моментах времени, и рассматривая систему только в эти моменты времени, мы получим дискретную динамическую систему, которую можно реализовать на клеточном автомате.

В данном разделе мы обсудим, как это делается. Некоторые детали не соответствуют реальной физике даже в большей степени, чем идеализированная физическая модель предыдущего раздела (например, шары будут иметь "диаметр" только вдоль направления движения). С другой стороны, другие детали более ралистичны; например, столкновения не являются моментальными: поскольку шары имеют протяженность в пространстве и дальнодействие запрещено, то при столкновении возмущение передается от одной части шара к другой за конечное время. На интуитивном уровне эту ситуацию можно описать так: упругость протяженного шара не может быть постулирована, а должна быть "синтезирована" на основе внутренних степеней свободы шара.

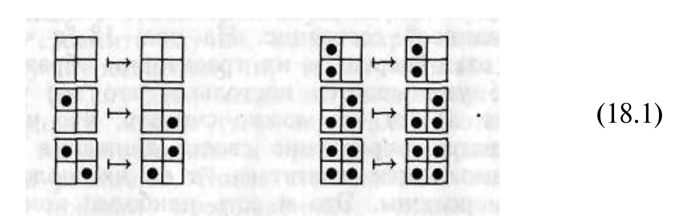
Так как модель биллиардных шаров - это по существу модель газа, то не удивительно, что наша реализация клеточного автомата тесно связана с моделями газа, которые мы обсуждали в предыдущих главах.[1] Вернемся к рассмотрению пра-

[1] Окрестность Марголуса первоначально была разработана как способ "сплутовать" и создать простую версию модели биллиардных шаров в виде клеточного автомата. И только позднее нами были разработаны для нее другие применения, такие, например, как модели газов.

вила SWAP-ON-DIAG разд. 12.2, в котором частицы двигаются по диагоналям массива клеток равномерно и без столкновений. Правило HPP-GAS, описанное непосредственно за SWAP-ON-DIAG (разд. 12.3), ввело в рассмотрение столкновения, при которых сохраняется импульс; такие взаимодействия сами по себе не подходят для наших целей, поскольку (а) они рассматривают частицы как имеющие нулевой диаметр и (Ь) они не обеспечивают некоторых эффектов "склеивания", на основе которых можно построить неподвижные зеркала (в самом деле, чтобы сделать контейнер для HPP-GAS, нам пришлось постулировать второй вид "вещества", поддерживаемый дополнительной плоскостью битов, как объяснялось в разд. 15.2).

Здесь мы к тому же используем взаимодействия, не сохраняющие импульс. *Заметим, что "шары" из модели биллиардных шаров не будут непосредственно отождествляться с элементарными частицами этой системы; вместо этого на основе этих частиц мы построим и шары, и зеркала как составные объекты.*

BBM-правило [35] использует окрестность Марголуса и представлено следующей таблицей:


(18.1)

Три первых элемента таблицы и последний идентичны соответствующим элементам HPP-GAS; два других элемента требуют более детального обсуждения.

Заметим, что содержимое блока изменяют только тогда, когда в нем находятся одна или две частицы. После определения отдельного слова языка Forth, которое занимается случаем только двух частиц, правило BBM легко выразить на CAM Forth:

```
                    : 2PART
          CENTER OPP = IF
                   CW ELSE
              CENTER THEN ;
                      : BBH
     CENTER CW CCW OPP + + +
   { U OPP 2PART U U } >PLNO ;
```

где и - сокращение для 'не изменяется" (т.е. CENTER), как в разд. 5.2.

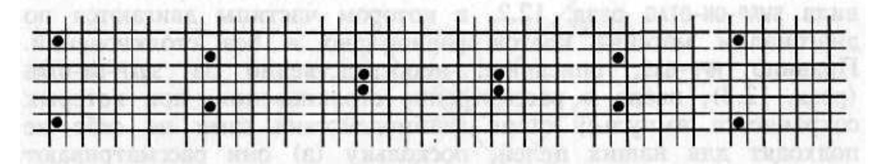


Рис. 18.4. Столкновение частиц под прямым углом: частицы разлетаются по своим предыдущим траекториям. Как на этой, так и на следующей диаграмме первый шаг всей последовательности шагов использует блоки четной сетки (жирные линии); второй шаг - нечетной сетки (тонкие линии) и так далее попеременно.

Зеркала. Сейчас мы обсудим действие четвертого элемента табл. (18.1). Если две частицы (двигающиеся, как и в HPP-GAS, по диагонали) испытывают столкновение под прямым углом, то после столкновения направление движения каждой из них меняется на противоположное, и частицы отскакивают назад по своим же следам (рис. 18.4). Такое взаимодействие (при котором импульс не сохраняется) позволяет группе частиц создавать "связанное" состояние. На рис. 18.5а четыре частицы отскакивают взад-вперед, и их траектории образуют ромб; когда этот ромб уменьшается настолько, что все частицы соприкасаются (рис. 18.5Ь), то можно считать, что частицы меняют на каждом шаге направление своего движения и при этом у них нет возможности двигаться, т. е. их положения в пространстве заморожены. Это и есть наиболее компактная форма *неподвижного зеркала* (заметим, что четыре частицы, образующие застывшую группу, охватывают два смежных блока). Частица, сталкивающаяся с зеркалом, также отскочит назад по своей траектории, как это определено пятым элементом табл. (18.1). Размер зеркала можно увеличить, располагая ряд зеркал впритык друг к другу (как, например, на рис. 18.7).

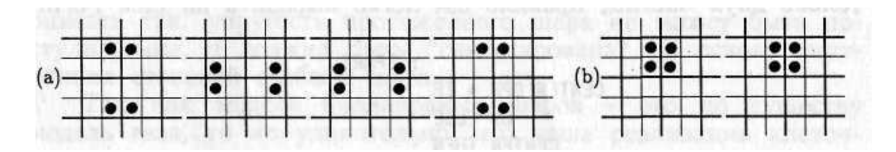


Рис. 18.5. Связанные состояния: (а) четыре частицы отскакивают взад-вперед по схеме ромба; (Ь) в случае их наиболее плотной упаковки шары слипаются и образуют неподвижное зеркало.

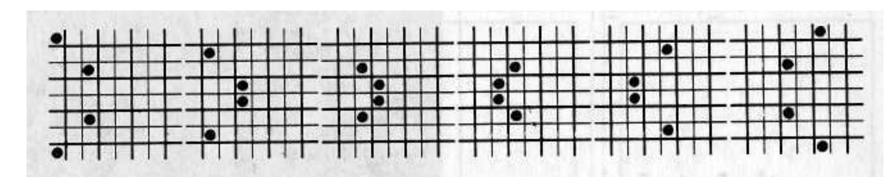


Рис. 18.6. Столкновение двух шаров; начинаем с нечетной сетки (тонкие линии).

Шары. Подобно зеркалам, "шары" модели биллиардных шаров - составные объекты. Они состоят из двух частиц, движущихся на фиксированном расстоянии друг от друга в одном направлении, по одной траектории. Столкновения под прямым углом между двумя шарами (рис. 18.6) представляют собой множественные столкновения между отдельными частицами, из которых состоят шары. При этом в тот или иной момент используются все четыре первых элемента табл. (18.1). Заметим, что когда речь идет о шарах (составных), то при их столкновении импульс сохраняется.

Если проследить траектории двух шаров, пути которых пересекаются, и сравнить случай, когда шары одновременно приходят в точку пересечения (и, таким образом, испытывают столкновение), со случаем, когда шары приходят в точку пересечения в разное время (т. е. не сталкиваются), то можно заметить, что геометрические и временные характеристики выходящих путей различны. После столкновения выходящие пути не являются прямым продолжением входящих путей (см. рис. 18.1); кроме того, при столкновении происходит задержка шаров. Это именно то, чего следует ожидать от "нежестких", хотя и упругих шаров.

Аналогичные рассуждения можно провести при столкновении шаров с зеркалами (рис. 18.7). Заметим, что зеркала ведут себя так^ как если бы имели бесконечную массу, и при такой интерпретации столкновения между шарами и зеркалами сохраняют импульс.

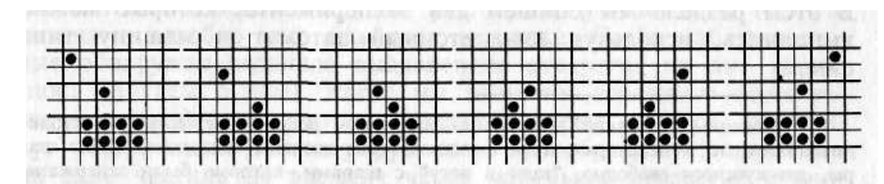


Рис. 18.7. Столкновение шара с зеркалом; начинаем с четной сетки (жирные линии).

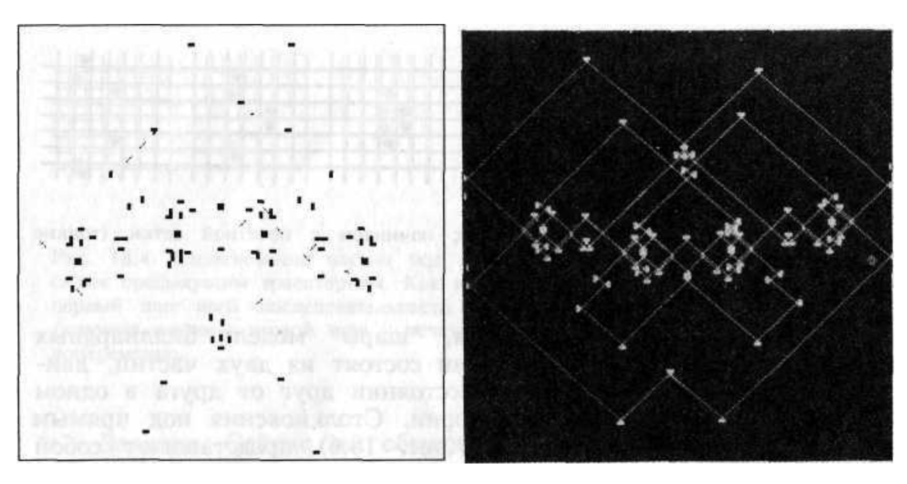


Рис. 18.8. Схема BBM-CA: (а) моментальный снимок и (Ь) снимок с продолжительной экспозицией.

Итак, мы создали структуру, которую на наиболее подробном уровне можно рассматривать как газ, состоящий из точечных частиц, взаимодействие между которыми происходит без сохранения импульса. На более абстрактном уровне, пока действуют некоторые ограничения на начальные условия, ту же самую структуру можно рассматривать как систему неподвижных зеркал и шаров конечного диаметра, динамика которых, с точки зрения физики, удивительно реалистична и вполне пригодна для реализации модели вычислений по методу биллиардных шаров. Рис. 18.8 представляет цифровую схему определенной сложности, реализованную на основе правила BBM.[1]

18.3. Несколько экспериментов с моделью биллиардных шаров

В этом разделе мы опишем два эксперимента, которые можно выполнить, используя BBM-клеточный автомат и машину типа CAM.

[1] Поскольку в такой реализации мы имеем дело с "нежесткими" столкновениями, то в некоторые пути сигналов были введены задержки, чтобы шары, движущиеся свободно, "шли в ногу" с шарами, которые были задержаны столкновением. Можно представить себе смысл этих задержек, если заметить, что число шагов, необходимых шару, чтобы проследовать по пути любого сигнала, в точности равно числу клеток, которые частица посетит на своем пути хотя бы один раз, включая и дополнительные клетки, где она побывает во время столкновения.

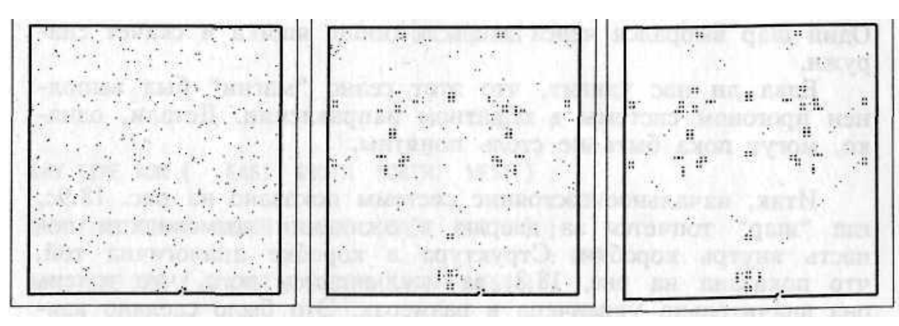


Рис. 18.9. Эксперимент с волшебным газом: (a) газ; (Ь) что-то происходит; (c) порядок возникает из беспорядка.

В отличие от необратимых правил, как, например, LIFE, ничего существенного не случится, если мы запустим BBM со случайного начального состояния: будучи газом, случайная конфигурация - это равновесная система, никакой дальнейшей эволюции не ожидается. Однако, когда вы сами создаете мир, вы можете устроить чудеса, как в первом эксперименте.

Когда правило клеточного автомата универсально с вычислительной точки зрения, то с помощью одного правила можно моделировать другое. Все же мы не должны забывать, что они не одно и то же, даже если они на первый взгляд кажутся таковыми; это проиллюстрировано во втором эксперименте.

18.3.1. Волшебный газ

Обладая обратимым компьютером, можно развлечь себя и удивить своих друзей, запуская его туда и обратно несколько раз. Однако было бы замечательно, если бы удалось сделать что-то действительно необычное. На рис. 18.9a у нас есть то, что выглядит подобно частицам газа в ящике. Особо наблюдательные обнаружат странное выдавленное отверстие на дне ящика (пока забудем о нем). Когда мы запустим эволюцию этого газа, мы обнаружим, что он некоторое время мечется по ящику, убедительно демонстрируя свою газообразность; при этом ни одна частица не вылетит через маленькое отверстие в днище ящика. Спустя примерно минуту после начала демонстрации начнут происходить странные вещи (рис. 18.9Ь): кажется, что газ начинает самоорганизовываться в нечто. Спустя еще

несколько секунд это преобразование завершается (рис. 18.9с). Один шар выбрался через дверь в днище ящика и скачет снаружи.

Едва ли нас удивит, что этот сеанс "магии" был выполнен прогоном системы в обратном направлении. Детали, однако, могут пока быть не столь понятны.

Итак, начальное состояние системы показано на рис. 18.9с, где "шар" топчется за дверью в ожидании возможности попасть внутрь коробки. Структура в коробке аналогична той, что показана на рис. 18.8, за исключением того, что теперь она значительно увеличена в размерах. Это было сделано взятием BBM-структуры обычного размера и равномерным увеличением расстояний между частицами. Благодаря масштабной инвариантности нашего правила (которой обладает HPP-GAS и другие аналогичные правила) такое преобразование приводит к новой конфигурации с изоморфным развитием: рассматриваемая через равномерные промежутки времени новая система - это расширенный вариант эволюции исходной системы. Зеркала расширенной системы не будут безусловно устойчивыми (в отличие от зеркал рис. 18.8). В самом деле, это - динамические объекты, целостность которых можно нарушить неправильным столкновением (см. рис. 18.5а и рис. 18.5Ь). Это, конечно, мы и имели в виду сделать.

Наш снаряд болтается снаружи у входа в ящик. Этот вход сконструирован очень тщательно. Чтобы пройти через него, требуется в точности правильный отскок, только шар вполне определенного размера способен сделать это. Отдельные частицы или даже пары с неправильным расположением просто будут отражены в направлении, обратном направлению подхода. Наш снаряд проникнет внутрь, но, поскольку он разрушает все, что находится внутри, шансы выйти наружу чему-нибудь в ближайшее время очень малы. Таким образом, мы можем дать возможность системе немного развиться; подождем, прежде чем прекратить моделирование, пока схема не распадется на частицы газа. В этот момент мы изменим направление движения всех частиц и запомним эту конфигурацию (показанную на рис. 18.9а), готовые к тому, чтобы удивить своих друзей.

18.3.2. **Конец** света

Пока ученик волшебника набирается опыта, волшебство может выйти из-под контроля. Сейчас мы проведем эксперимент, в котором чудо приведет к концу света.

```
NEW-EXPERIMENT  N/MOORE  &/HV
                                   0 CONSTANT TIME

                                                : U
                                        CENTER ;
                                                : CW
Ш TIME XOR {   EAST  SOUTH  NORTH   WEST } ;
                                                : CCW
Ш TIME XOR {  SOUTH   WEST   EAST  NORTH } ;
                                                : OPP
&HV TIME XOR { S.EAST S.WEST N.EAST N.WEST } ;
                                                : 2RUL
         CENTER OPP = IF CW ELSE CENTER THEN ;
                                                : BBRUL
  CENTER CW CCW OPP + + + { U OPP 2RUL U U } ;
                                                : EOW
     0 IS TIME BBRUL
     3 IS TIME BBRUL  XOR CENTER^1 XOR  >PLNO
                                  CENTER  >PLN1 ;
MAKE-TABLE EOW
```

Вот что мы сделали: мы написали BBM в виде обратимого правила второго порядка (см. разд. 12.5 и 14.2). Это законченное определение эксперимента - заметим, что такая версия правила работает с окрестностью N/MOORE и совсем не использует чередующихся сеток. Как это может быть?

Назовем $г_e$ преобразование некоторой конфигурации, произведенное правилом BBM при использовании четной сетки. На четном шаге конфигурация c^t перейдет в конфигурацию $c^{t+1}=m_e c^t$. Поскольку m_e совпадает с обратным к нему преобразованием, то мы можем также написать $c^* = r_e c^{t+1}$. Аналогичные рассуждения применимы к преобразованию $т_о$, выполняемому правилом BBM при использовании нечетной сетки.

Пусть даны три последовательные конфигурации c'"[1], c^*, $_c$f+i. тогда независимо от четности первого выполняемого шага имеет место следующее равенство:

$$T_e c^t + r_o c^t \quad = \quad c^{t-1} + c^{t+1}$$

(сумма двух конфигураций - это конфигурация, полученная сложением по модулю 2 соответствующих клеток исходных конфигураций); другими словами, складывая результат выполнения шага вперед с результатом выполнения шага назад, получают будущее плюс прошедшее.

Далее, можно определить новое преобразование $m = m_e + m_o$, так что

$$mc^* = c^{t+1} + c^{*\prime\prime},$$

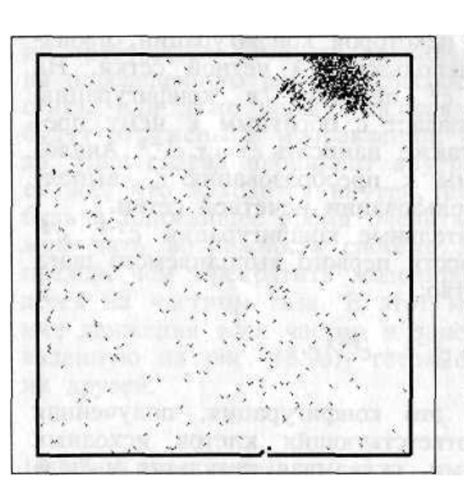
или

$$<\Gamma-\backslash_e m ?-<\Gamma^*. \qquad (18.2)$$

Но это - обратимое правило второго порядка в форме, которую мы уже видели в разд. 14.2. Чередование сеток исчезло, поскольку мы складывали вместе четные и нечетные шаги.

Если мы сейчас запустим такое правило второго порядка из состояния рис. 18.9а (с включенным ECHO), то обнаружим, что эволюция продолжается так, как если бы мы использовали правило ВВМ Мы даже можем инвертировать систему, поменяв местами данные настоящего с данными прошедшего.

А теперь мы действительно покажем чудо. Изменим значение клетки в настоящем без соответствующего изменения в прошлом. Это представлено на рис. 18.10а и 18.10Ь. Газ, который выглядит вполне случайным, на самом деле был все еще весьма организованным по сравнению с действительно случайным газом. Большинство состояний такой системы второго порядка не соответствуют какому-либо состоянию газа ВВМ Когда мы создали наше чудо, то в тонкой ткани нашего ВВМ моделирования возникла дыра, и мир разлетелся на куски.



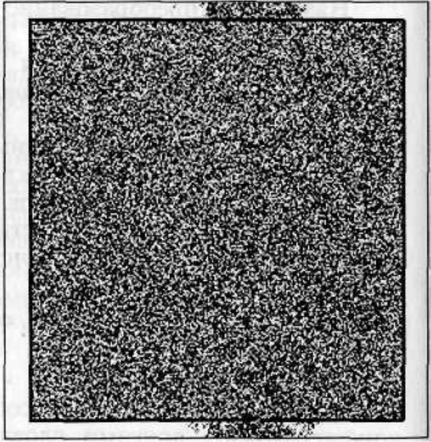


Рис. 18.10. Конец мира: (а) ткань начинает распускаться; (Ь) конец.

ВЫВОДЫ

Клеточные автоматы позволяют создавать сложные объекты из простых материалов. В этом смысле они ближе по духу к математическим моделям, используемым в достаточно сложных областях теоретической физики, чем к более простым моделям, используемым в вычислительной физике.

К тому же они представляют собой нечто большее, нежели полезные абстракции. Клеточные автоматы обладают двумя действительно фундаментальными достоинствами, которые могут привести к исключительно практичным архитектурам компьютеров.

Во-первых, они по своей природе параллельны. Если мы свяжем один процессор с каждыми N клетками, то мы можем увеличивать размеры моделей неограниченно, не увеличивая времени, необходимого для завершения обновления пространства. Нет неизбежных накладных расходов, связанных с разбиением задачи по многим процессорам или с координацией деятельности большого числа процессоров. Поэтому модели на основе клеточных автоматов допускают массивные реализации, ограниченные не архитектурой, а экономическими соображениями.

Во-вторых, клеточные автоматы по сути своей локальны. Из-за ограниченности скорости света локальность соединений между простыми исполнительными элементами отражается на скорости выполнения операции. В обычных компьютерах время исполнения одной команды ограничено наибольшим путем прохождения сигнала, и поэтому быстродействующие компьютеры должны быть компактными. В клеточных автоматах размеры компьютера не зависят от длины пути прохождения сигнала и поэтому машины могут быть одновременно и очень большими, и очень быстрыми.

Оба эти фактора, внутренне присущие параллелизм и локальность, - следствие того, что клеточные автоматы в действительности являются стилизованными моделями физики, обладающие такими особенностями, как реалистичные время и пространство. Поэтому их можно более непосредственно, чем другие архитектуры, отобразить на физические реализации. Поэтому они также хорошо подходят и для различных задач физического моделирования, как мы обсуждали в данной книге.

16-1608

Все это заставляет нас верить, что клеточные автоматы станут основой важных параллельных архитектур. Выбор конкретной формы архитектуры зависит от сложного компромисса, включающего такие факторы, как технологические возможности, сравнительная важность требований к скорости и к масштабам моделирования, сложность операций в каждой клетке, объем ввода/вывода, необходимый для проверки и изменения значений переменных, включение нелокальных (или менее локальных) черт в целях увеличения общности.

Ясно, что разработка полезных моделей, в которых нуждаются такие машины, была и продолжает быть важным стимулом к их развитию. И наоборот, перспектива небывалого увеличения скорости и полноты моделирования необычайно способствует дальнейшей разработке моделей и технологий моделирования, которые используют мощь таких машин.

Приложение А

КРАТКОЕ РУКОВОДСТВО ПО ЯЗЫКУ FORTH

Основная цель этого руководства состоит в том, чтобы читатель мог *читать* тексты на языке Forth достаточно свободно для понимания примеров программирования машин клеточных автоматов (CAM), приведенных в данной книге.

Для организации полномасштабных экспериментов с CAM необходим относительно ограниченный объем знаний об этом языке: одни и те же немногочисленные конструкции языка Forth повторяются вновь и вновь с незначительными вариациями; большинство же специфических возможностей этого языка, играющих важную роль в других приложениях, здесь не понадобится.

Онако эти несколько конструкций должны быть хорошо поняты. Во многих случаях достаточно интуитивного представления, но мы без колебаний будем приводить нужное количество технических подробностей в тех немногих случаях, когда необходимо гарантировать точное понимание.

А,1. Интерпретатор команд

Вы можете мысленно представить себе *интерпретатор команд* Forth как компетентного, но не слишком разговорчивого специалиста, который сидит в машинном зале вашего компьютера и имеет доступ ко всем органам управления и измерительным приборам. С пульта управления вы разговариваете с ним через "селектор", т.е. через терминал, отдавая приказы и получая ответы и подтверждения. Например, если вы скажете

```
BEEP
```

(напечатав это на экране терминала и нажав клавишу "ВОЗВРАТ КАРЕТКИ" ("ВК")), терминал ответит подачей звукового сигнала "БИП"; если вы скажете

```
0 100 DUMP
```

то содержимое ста ячеек памяти (начиная с нулевой) будет выведено на экран. После этого интепретатор скажет

```
OK
```

чтобы сообщить вам, что он все исполнил и ждет дальнейших указаний. (Здесь и далее мы полагаем, что вам ясны смысл выражения OK и назначение клавиши "ВК").

"Уши" интерпретатора устроены так, что могут разбивать входной поток символов на фрагменты, используя "пустое пространство" (одиночный или подряд идущие пробелы) в качестве разграничителей фрагментов. Эти фрагменты последовательно друг за другом проходят через "мозг" интерпретатора, пробелы отбрасываются. Таким образом, интерпретатор "услышит" предыдущую команду как последовательность трех фрагментов - о, 100, DUMP - и "услышит" то же самое, если команда будет, например, такой:

```
0 100          DUMP
```

Для того, чтобы интерпретатор вас понял, и вы, и он должны использовать один и тот же словарь терминов и придерживаться разнообразных соглашений, которые вместе со словарем и составляют язык Forth. Содержимое словаря отображает ряд фактов, о которых знает интерпретатор на данный момент. Выдав команду, вы обнаружите, что интерпретатор уже прошел "стандартное обучение", а, может быть, и более специализированное обучение (например, как управлять САМ). Это стандартное обучение, документированное в любом хорошем руководстве по языку Forth, играет в языке важную роль, что выгодно отличает Forth от других языков программирования; и в этом смысле о нем говорят как о *среде программирования,* а не просто как о *языке программирования.*

k.l. Компилятор

Элементы, из которых состоит словарь, называются, как и следовало ожидать, *словами.* Процесс *программирования* на Forth состоит в последовательном добавлении вами новых слов в словарь, причем новые слова определяются через уже существующие. Таким путем вы расширяете познания интерпретатора (а также и свои выразительные средства) в интересующей вас области деятельности. На любом этапе разработки вы можете сказать нечто, включающее новые слова, и проверить, что действительное значение этого выражения (т. е. что интерпретатор делает в ответ на ваши слова) - это то, что вы имеете в виду.

Например, чтобы образовать новое слово для действия "подать сигнал "БИП" три раза", вы печатаете на экране

```
: 3BEEP  BEEP BEEP BEEP ;
```

Как только интерпретатор обнаружит двоеточие Y, он призывает на помощь другого специалиста - компилятор определений COLON. Этот специалист берет фрагмент, непосредственно

следующий за Y (в нашем примере 3BEEP) и создает под этим именем новую запись в словаре. Далее он ожидает фразу, которая описывает действие нового слова; эта фраза - BEEP BEEP BEEP - в данный момент не исполняется, а компилируется в словарь как значение слова 3BEEP. Завершает фразу символ V, который говорит компилятору, что нужно вернуть управление интерпретатору.

Язык, понятный компилятору COLON, слегка отличается от (и несколько богаче) языка, понятного интерпретатору команд. Только при отработке некоторых команд компилятор и интерпретатор не взаимодействуют друг с другом. В дальнейшем такие команды нам не понадобятся.

Если вы напечатаете на экране

```
3BEEP
```

интерпретатор найдет это слово в словаре, исполнит его и выдаст в ответ OK. А если вы напечатаете

```
4BEEP
```

интерпретатор будет искать, но не найдет этого слова в словаре; тогда он спросит у своего помощника, является ли данное слово числом (см. следующий раздел); и, наконец, напечатает

```
4BEEP ?
```

сообщая тем самым, что фрагмент 4BEEP не имеет смысла.

А.3. Словарь

Если вы напечатаете на экране FORTH WORDS, в ответ получите список всех слов, находящихся на данный момент в основном разделе Forth-словаря[1], причем в начале списка стоит слово, введенное последним по времени. Таким образом, если предыдущий пример взаимодействия с интерпретатором был исполнен, то слово 3BEEP окажется в начале списка. Более раннее слово BEEP стоит в списке ниже.

Следующие записи
(

```
BEEP HERE + ' CONSTANT C@ ; : C! 0=
```

взятые произвольно из словаря, дают представление о том, как выглядят обычные Forth-слова. Любой фрагмент может быть помещен в словарь как слово. В частности, символы, используемые в других языках как знаки препинания, например, Y, могут быть частью Forth-слова или даже самостоятельным

[1] Кроме этого основного раздела, называемого FORTH, словарь может содержать некоторые специальные разделы, что мы вскоре объясним.

Forth-словом; слово 'C," существенно отличается от последовательности из двух слов 'с ,\
Можно переопределить ранее определенное слово. Например, если динамик вашего компьютера вышел из строя, в качестве временного выхода из положения можно использовать новую версию BEEP, которая с новой строки экрана выводит сообщение 'Верите или нет, но это "БИП"!\ Переопределим BEEP следующим образом:

```
: BEEP CR ." Верите или нет, но это 'БИП'!" ;
```

(конструкция ."<текст>" - команда напечатать <текст> на экране - может присутствовать только внутри COLON-определений (от : - colon). Если определить

```
: 4BEEP BEEP BEEP BEEP BEEP;
```

то данное слово скомпилируется с использованием новой версии BEEP и при исполнении напечатает

```
Верите или нет, но это 'БИПМ
Верите или нет, но это 'БИП'!
Верите или нет, но это 'БИП'!
Верите или нет, но это 'БИП'!
```

Старая версия BEEP остается в словаре, и ранее определенное слово 3BEEP *сохранит свое первоначальное значение,* связанное со старой версией BEEP. Если теперь исполнить 3BEEP, то три сигнала будут посылаться на (испорченный) динамик.

Вполне законно и часто полезно переопределить некоторое слово, используя в определении его же. Например, если звуковой сигнал вашего терминала имеет такое время затухания, что три последовательных сигнала сливаются в один, то можно переопределить BEEP так:

```
: BEEP  BEEP 10 TICKS ;
```

(10 TICKS означает "Ждать 10 тактов внутренних часов компьютера"), и последовательность сигналов теперь будет звучать раздельно, а не слитно.

В некоторых словарях естественных языков слова, относящиеся к некоторым специальным областям, группируются в отдельных разделах (например, географические названия, единицы измерений, сокращения). В языке Forth эти разделы называются "vocabularies" (по-русски - словарь, лексикон); в дополнение к главному FORTH-словарю существует ASSEMBLER-словарь, содержащий машинные коды операций и другие ассемб-

лерные понятия, EDITOR-словарь, содержащий команды редактирования, и т.д.

Поскольку поиск слов производится лишь в текущем "активном" словаре, разбивка на разделы позволяет использовать одинаковые имена с различными значениями в разных контекстах или делает слово недоступным в определенном контексте. В CAM Forth определенные слова, характеризующие отношение соседства (S.EAST, &CENTER и др.), выделены в специальные разделы для того, чтобы сделать их доступными только тогда, когда соответствующие слова соседства подсоединены к справочной таблице.

А.4. Числа

Если в каком-либо обычном тексте на естественном языке встретится фраза "предложение прошло 371 голосом "за" ", то нет нужды искать фрагмент "371" в словаре, да вы там его и не найдете. Значение числа вытекает из его строения; по этой причине кто угодно может создать больше чисел, чем комулибо захочется перечислить, но по этой же причине нет необходимости заносить значение каждого числа в словарь.

Аналогичная ситуация и в языке Forth: грамматический разбор чисел производится по мере их поступления; значение числа формируется исходя из его строения еще одним специалистом - NUMBER (число), вызываемым интерпретатором в случае необходимости. "Значение" - не что иное, как внутреннее представление в двоичном виде. Например, если интерпретатор обнаруживает фрагмент 100 и работает в DECIMAL (десятичный) режиме, то фрагмент будет распознан как число и преобразуется во внутреннее представление 0000000001100100 (Forth хранит целые числа в 16-битовых ячейках); однако, если поменять режим интерпретатора на HEX (16-тиричный) режим,[1] то тот же самый фрагмент получит иное значение, а именно 0000000100000000.

Некоторые числа, в силу исторических или практических причин заслуживающие явного упоминания, скажем, "three", все-таки включены в обычный английский словарь. Аналогичным образом такие числа, как 0 и 1, помещены (именно "0", а не ZERO) в качестве *слов* в Forth-словаре.[2] Это ведет к более эффективному исполнению (их значения устанавливаются заранее раз и навсегда, и нет необходимости обращаться за помощью к специалисту NUMBER) и более компактному коду.

[1] То есть в системе счисления с основанием шестнадцать, а не десять. В языке Forth числа можно читать и записывать в системе счисления с произвольным основанием по вашему выбору.

[2] Слова типа CONSTANT; см. разд. А.8.

А.5. Стек

Forth преуспел в достижении замечательной выразительной мощи и эффективности; кроме того, *система* Forth (т. е. реализация языка на компьютере) может быть изумительно проста и компактна. Эти преимущества получены за счет жесткой стандартизации, в частности, тем, как соседние слова фразы передают одно другому информацию, связывающую их в единое синтаксическое целое.

Для такой передачи все данные (символы, булевы переменные, числа, адреса и проч.) упаковывают в "коробку" стандартных размеров, называемую *ячейкой,* состоящую из 16 бит. Все обмены данными происходят (по заранее определенному сценарию, как мы это сейчас увидим) в единственном банке - *стеке.* Он представляет собой стопку ячеек, которая растет или сокращается в зависимости от поступления новых данных.

Представим себе сцену, на ее середине - стек. Интерпретатор Forth - хореограф; по мере того, как он считывает слова, составляющие вашу фразу, на сцену по очереди выходят соответствующие актеры, делают свое дело и исчезают. Если роль требует от актера оставить некоторые данные для последующих актеров, то он идет к стеку и укладывает эти данные, ячейка за ячейкой, друг на друга; если же в его роль входит получение данных от предыдущего актера, то он подходит к стеку и забирает их.

Некоторые необходимые актеру данные могут лежать в глубине, скажем, ниже на две ячейки от вершины стека. Актеру не нужно в поиске этих данных нервно рыться в стеке (коробки не имеют наклеек!). Действительно, партитура просто подскажет ему, что нужно поднять две верхние ячейки, взять ячейку, которая теперь стала верхней, и положить те две обратно.

По этой схеме каждому элементу данных нет необходимости иметь свой *абсолютный* адрес - долговременный почтовый ящик с известным именем. Вместо этого адресация осуществляется положением *относительно* вершины стека. Если новый, независимый элемент сценария вставить старую партитуру, то в момент его исполнения можно обнаружить, что стек то слегка вырастет, то чуть-чуть сократится и так далее, и в конце вставленного элемента вернется к своей начальной высоте. Остальная часть партитуры будет продолжена, причем данные для нее остались там же, где и были оставлены.

А.6. Выражения

Стековая дисциплина хорошо приспособлена к требованиям передачи данных внутри иерархически построенной программы. Она позволяет использовать очнь простую счетную запись - польскую инверсную запись - посредством которой арифметические и логические выражения произвольной глубины можно записать без помощи скобок и других маркеров положения.

Если напечатать число интерпретатору команд, то оно упаковывается в одну ячейку и заносится на вершину стека. Односимвольное слово "." ("точка") снимает верхнюю ячейку стека и выводит ее содержимое на экран в качестве числа. Таким образом, если напечатать

```
356 .
```

(здесь "точка" - часть того, что вы печатаете), то на экране появится

```
356
```

(В более обычных языках программирования эквивалентом для выражения "356 ." может быть что-нибудь вроде "pr1nt(356)".) Заметим, что при вводе 356 стек вырос на 1, а при вводе точки - понизился на 1, т. е. сохранил свою высоту.

Слово "+" ("плюс") хватает *две* верхних ячейки стека, складывает их и помещает результат *(одну* ячейку) на вершину стека. Таким образом, слово "+" оставляет стек опущенным на один уровень по отношению к началу выполнения сложения. Например, выражение

```
2 3 +
```

оставляет в результате на вершине стека число 5 (откуда его можно вывести на экран посредством "."). Если интересно знать, сколько будет 1+2+4, наберите на клавиатуре

```
1 2 + 4 + .
```

Докажем эволюцию стека на бумаге:

СТЕК	ВХОД	ВЫХОД
	1	
... 1	2	
... 1 2	+	
... 3	4	
... 3 4	+	
... 7	.	7

где каждый ряд отражает: а) текущее состояние стека (верхний элемент стека - справа); Ь) текст, который будет интерпретирован; с) то, что отображается на экране. Точки слева обозначают часть стека, которую мы не трогаем. В следующих примерах со стеком они будут опущены.

Заметим, что если число 3 было помещено на стек, то не имеет значения, *как* оно туда попало; с функциональной точки зрения выражение 1 2 + равнозначно, скажем, 3 или 1 1 + 1 +, или чему-нибудь еще, что в конце концов сошло со сцены, оставив 3 на вершине стека. Заметим еще, что два различных выражения

1 1 + 1 + 1 +   и   1 1 1 1 + + +

дают одинаковый результат, хотя второе временно поднимает стек до более высокого уровня:

```
     СТЕК      ВХОД           СТЕК      ВХОД
                 1                        1
      1    1                        1     1
      11         +              11        1
      2          1               1  1  1  1
      2 1        +              1 1 1 1   +
      3              1                112+
      3 1     ,  +               13       +
      4                         4
```

А.7. Редактирование и загрузка

Если интерпретатору уже выдана команда, то ее невозможно отменить; при возникновении ошибки даже можно и не вспомнить выданную команду. Хотя прямое общение с интерпретатором весьма полезно, бывают моменты, когда хочется заранее тщательно обдумать всю последовательность команд и определений, просмотреть и отредактировать ее, и, возможно, обсудить ее с кем-нибудь еще перед выдачей интерпретатору. Хочется также иметь возможность отдавать *заранее написанные* приказы или даже иметь набор различных "списков распоряжений", чтобы выдавать их интерпретатору в зависимости от обстоятельств.

Все это можно сделать, записав сначала текст в файл на диске, где его можно просмотреть и изменить средствами *экранного редактора* Forth. Затем можно попросить интерпретатор использовать этот файл (или часть файла) в качестве входного потока вместо того, который идет от клавиатуры;

этот процесс называется *загрузкой*. Когда вы загружаете файл, все происходит так, как будто содержимое файла вы набираете с клавиатуры - за исключением того, что интерпретатор отрабатывает фрагменты при их поступлении без ожидания появления "возврата каретки".[1]

Создавая с помощью редактора текст для последующей загрузки, его можно для облегчения понимания отформатировать и добавить, если хочется, в нужных местах собственный комментарий.

Схема формата, принятая в нашей книге, выглядит так:

```
                  : 3BEEP
      BEEP BEEP BEEP ;
                      : BEEP
              BEEP
          10 TICKS ;
                      : 4BEEP
  BEEP BEEP BEEP BEEP ;
```

Здесь элементы словаря выровнены на правой половине страницы, а "тела" этих определений - на левой.

Forth-слово "(" удаляет из входного потока все, что следует до соответствующего ему ")", включая и сам этот символ; так, можно писать

```
BEEP BEEP ( два бипа ) BEEP BEEP ( еще два )
```

и интерпретатор никогда не узнает, что находится "в скобках".[2]

Слово "\" ("обратная косая черта") рассматривает в качестве комментария остаток строки, в которой появляется "\".

```
              : BEEP   \ Новая версия
         BEEP          \ просто бил
     10 TICKS ;        \ вставляет задержку
```

[1] Типичный исходный Forth-файл совсем не содержит "ВК"; строки, которые вы видите при редактировании на экране, запоминаются в файле друг за другом без каких-либо добавлений разделительных знаков.

[2] Заметим, что следующая строка и положения пробелов в ней корректны:

```
BEEP BEEP ( два бипа)BEEP BEEP ( еще два),
```

хотя и нет разделяющего пробела между ")" и **BEEP,** поскольку действие слова ")" состоит именно в том, что оно удаляет строку **"два бипа)".**

С другой стороны, разбиение

```
BEEP BEEP (два бипа) BEEP BEEP (еще два)
```

некорректно, поскольку оно заставит интерпретатор рассматривать **"(два"** как символ, который нужно отработать.

А.8. "Константы" и "переменные"

В разд. А.2 мы говорили, что интерпретатор "исполняет" слова, которые печатаются. На самом деле каждое слово в Forth-словаре несет пометку "я должно быть исполнено таким-то специалистом, который знает, как меня исполнить", а интерпретатор лишь передает это дело указанному специалисту. Для слов, которые были введены в словарь COLON-КОМПИЛЯТОром, знающий специалист - это *COLON-интерпретатор.*

В общем, у каждого типа слов есть свой собственный компилятор и соответствующий ему интерпретатор. Если слово скомпилировано ассемблером, то дальнейшая передача ответственности прекращается; этот специалист порождает код, написанный непосредственно на машинном языке (т. е. на родном языке вашего микропроцессора), и здесь за дело берется аппаратура.

Все это звучит гораздо сложнее, чем работает. Предположим, вы пишете программу управления телескопом, которой необходимо знать широту вашего города, скажем, 43 градуса. Считается хорошим стилем программирования присвоить этому числу *имя* - скажем, LATITUDE - с тем, чтобы где бы оно ни появилось в программе, эффект был бы тот же, как если бы вы напечатали "43" на экране. Чтобы сделать это в Forth, вы пишете

```
43 CONSTANT LATITUDE
```

Слово LATITUDE будет занесено в словарь как константа, и при исполнении оно положит число 43 на вершину стека.

А происходит при этом вот что. Как только встретится слово CONSTANT, интерпретатор обращается за помощью к CONSTANT-компилятору, который хватает следующий фрагмент - LATITUDE - и заводит новый элемент в словаре под этим именем. Элемент состоит из двух частей: первая *(область кода)* содержит пометку "я должен быть исполнен CONSTANT-интерпретатором"; вторая *(ячейка данных)* - резервируется для величины константы. В этом месте CONSTANT-КОМПИЛЯТОР берет верхнюю ячейку стека - с числом 43, которое было только что туда положено - и переносит ее в ячейку данных в словаре. CONSTANT-интерпретатор во время исполнения просмотрит ячейку данных и ее копию положит на вершину стека.

Учитывая предыдущее определение для LATITUDE, выражение

```
LATITUDE 7 + .
```

выведет 50 на экран.

Термин "CONSTANT" употребляется, в общем-то, неправомерно (хотя и используется в силу исторических причин), поскольку содержимое ячейки данных по желанию может быть изменено; в данной реализации языка Forth, чтобы заменить значение LATITUDE на 45°, нужно написать

```
45 IS LATITUDE
```

Самый важный аспект константы в языке Forth состоит в том, что она возвращает *значение* соответствующей ячейки данных, а не *указатель* на это значение (см. VARIABLE ниже).

В САМ слова, означающие отношения соседства - NORTH, SOUTH и т. д. - действуют подобно константам, поскольку они возвращают значение; это значение не один раз изменится во время построения таблицы для правила.[1]

Forth предоставляет еще одну возможность доступа к элементу данных, а именно через *адрес* этого элемента.[2] Так, VARIABLE-КОМПИЛЯТОР, используемый в конструкции

```
VARIABLE TIME
```

аналогичен CONSTANT-компилятору в части создания элемента словаря - TIME - с ячейкой данных в нем. Однако:

- Эта ячейка данных не загружается начальным значением (и поэтому определяющее слово VARIABLE, в отличие от CONSTANT, *не* ожидает значения на вершине стека).

- При выполнении TIME VARiABLE-интерпретатор кладет на стек адрес ячейки данных, а не ее содержимое.

[1] Вы пишете правило для САМ в виде Forth-слова (скажем, LIFE), определяющая фраза которого будет, конечно, использовать слова, обозначающие отношения соседства. Когда слово MAKE-TABLE создает для вашего правила справочную таблицу (см. разд. 4.2), оно исполнит слово LIFE для каждого элемента справочной таблицы, и слова, означающие отношения соседства, будут задействованы. Эти слова отличаются от обычных констант тем, что все они используют единственную ячейку данных. При создании и-го элемента таблицы "номер элемента" *п* храниться в этой ячейке; когда вызывается слово, обозначающее отношение соседства, оно проверяет номер элемента и возвращает соответствующее значение отношения соседства для данного элемента. Если по какой-то странной прихоти вы решили вмешаться в содержимое ячейки "номер элемента" во время создания таблицы, то значения, возвращаемые *всеми* словами, обозначающими отношение соседства, будут затронуты.

[2] В обычных компьютерах ячейки памяти нумеруются последовательно; адрес элемента данных - это номер ячейки, в которой находится элемент данных.

Поэтому если мы наберем на клавиатуре

```
TIME
```

то на стек будет помещено не текущее значение переменной TIME (которое может несколько раз измениться в ходе выполнения программы), а ее адрес (который всегда один и тот же).

Чтобы получить *значение* переменной TIME, нужно использовать слово "@", к

```
TIME 8 ( данные )
```

(т. е. "8" ожидает адрес на стеке и заменяет его данными, находящимся в памяти по этому адресу). Чтобы изменить значение переменной TIME, нужно использовать слово "!" ("запомнить"), как в

```
( данные ) TIME !
```

(т. е. "!" ожидает в стеке и элемент данных, *и* адрес, и заносит данные по адресу). Например, чтобы увеличить значение переменной TIME на 1, пишем

```
TIME 0 1 + TIME !
```

Предположим, что ячейка данных для TIME расположена в адресе 1000 и что ее начальное значение равно 5, тогда изменение этого значения будет происходить следующим образом:

Содержимое TIME	СТЕК	ВХОД
5		TIME
5	1000	@
5	5	1
5	5 1	+
5	6	TIME
5	6 1000	!
6		

С точки зрения пользователя САМ, Forth-VARIABLE придется применять редко, если вообще придется. В нашей книге термин "переменная" всегда употребляется в обычном смысле - "типовая величина, которой можно присвоить произвольное значение", а не более специальном смысле, который оно имеет в языке Forth.

А.9. Итерация

Если программа хранится в памяти компьютера, то отдельные ее части с незначительными изменениями можно использовать снова и снова, вводя при этом, если нужно, некоторые изменения. Например, можно определить

```
: BEEP-STUCK
   BEGIN
  BEEP
   AGAIN ;
```

При вызове этого слова, если выполнение достигло AGAIN, то оно возвращается к BEGIN, производя бесконечную серию "бипов". Приходится выключать питание, иначе из этого цикла не выйти.[1]

Упомянутая пара BEGIN/AGAIN ограничивает фразу подобно паре скобок: фраза внутри этих "скобок" будет повторяться бесконечно. Заметим, что для удобства прочтения мы выровняли оба элемента пары по вертикали по правому краю (это рекомендуется делать при польской инверсной записи), а внутреннюю фразу расположили с отступом от этой вертикали.

Более гибкой парой является пара DO/LOOP, слово

```
: 100BEEPS
100 0 DO
  BEEP
    LOOP ;
```

будет производить сигнал "бип" 100 раз.[2] А происходит это следующим образом: слово DO хватает два числа с вершины стека - 100 (число повторений цикла LIMIT) И 0 (начальное значение индекса цикла INDEX) - и сохраняет их для дальнейшего использования. Исполнение продолжается до тех пор, пока не встретится слово LOOP. В этот момент индекс увеличивают на 1 и сравнивают с числом повторений цикла: если эти величины равны, то цикл завершен, в противном случае исполнение возвращается на слово DO.[3]

[1] Это не совсем так, если ваш компьютер располагает клавишей BREAK, эквивалентной "паника-клавише".

[2] Конечно, если бы вы напечатали 999BEEPS, у вас тоже получилось бы слово, которое заставляет компьютер 100 раз делать "бип". Имя - это всего лишь имя...

[3] Индекс цикла берется по модулю 2^{16}. Минимальное число итераций 1 достигается тогда, когда цикл начинается с индекса, который на 1 меньше числа повторений; максимум итераций - когда индекс равен числу повторений: 0 0 LOOP будет выполняться 2^{16} раз!

Пары "скобок" BEGIN/AGAIN, DO/LOOP могут быть вложены друг в друга как обычные скобки, и тогда можно звонить и свистеть сколько хочешь. Вы можете более детально ознакомиться с этими и другими конструкциями управления программой в руководстве по Forth. Здесь же мы только упомянем, что конструкции управления могут появляться только в COLON-определении. Вы не имеете права говорить

```
100 0 DO BEEP LOOP
```

на уровне интерпретатора команд.

А.10. Стековые комментарии

К этому моменту мы уже не раз сталкивались со словами, ожидающими аргументов на стеке или возвращающих результаты в стек. Поскольку нарушение стековой дисциплины делает вычисления неуправляемыми (если вы оставили лишний элемент в стеке, то кто-то после вас получит неправильные данные), то было бы удобным иметь обозначения, показывающие, сколько данных слово забирает из стека и сколько в нем оставляет.

Ниже приведены примеры *стековых комментариев:*

```
DO         ( nl  n2  — )
DO          ( limit index -)
BEEP        ( -)
LATTITUDE   ( -n)
TIME        ( - addr)
+          ( nl  n2  —  n3)
+           ( m n  —  m+n)
2          ( -n)
2          ( ~2)
```

Примем следующее соглашение: помещаем в скобки "тире" (обычно двойное тире) для указания рассматриваемого слова; перед тире пишем список того, что данное слово *ожидает* найти в стеке, а после тире - список того, что данное слово *оставляет* в стеке.

Существенно лишь число элементов в списке - оно соответствует количеству ячеек, снятых со стека или положенных на стек; над сами ми же элементами можно слегка поработать, добиваясь большей ясности.

Например, слово DO берет два элемента из стека и ничего не возвращает. Минимальные обозначения дают

```
DO ( nl n2 - )
```

они лишь говорят, что DO ожидает два элемента в стеке. Большую ясность несет обозначение

```
DO ( limit index --)
```

которое подсказывает, что первый элемент используется в качестве числа повторений цикла, а второй (его начальное значение) как *индекс* цикла.

И напоследок приведем превосходный пример загрузки следующих трех определений из файла на диске:

```
  100 CONSTANT HUNDRED ( ~ п)
10000 CONSTANT A-LOT-OF ( - п)
: BEEPS ( - п)
0 D0
BEEP LOOP ;
```

Затем наберем команды

```
3 BEEPS
HUNDRED BEEPS
A-LOT-OF BEEPS
```

Выше мы видели, что DO требует двух аргументов. Когда мы пишем 3BEEPS, то первый аргумент посылается в стек числом 3, в то время как второй - числом о, находящимся в определении слова BEEPS. Итак, требования слова D0 удовлетворены.

Заметим, что если определять слова, привносящие некоторый смысл в "стековое взаимодействие" *(кто* должен передать или получить, *что* и *когда),* то изображению Forth-фразы можно придать подобие фразы естественного языка, чего очень трудно достичь в других языках программирования. Соответствующим образом приготовленные Forth-слова могут разговаривать друг с другом внутри фразы, совершенно не беспокоя нас своей болтовней.

В Forth есть соглашение, которого необходимо придерживаться: старайтесь создавать слова, снимающие аргументы со стека, а не оставляющие их в стеке. Если же элемент стека используется как аргумент двумя соседними словами, то перед тем как взять его из стека, необходимо сделать копию этого элемента посредством слова DUP, которое описывается в следующем разделе.

А.11. DUP, DROP и т. д.

Forth-слово "*" (умножить) берет два числа и возвращает произведение этих чисел; чтобы вычислить квадрат числа 3, надо напечатать число 3 дважды: з 3 * . А как насчет слова SQR, которое берет единственный аргумент и умножает его на себя же?

Forth имеет в своем распоряжении универсальные слова для работы со стеком; одно из них DUP (произносится как "dupe"), которое просматривает верхнюю ячейку стека и на нее помещает ее же *копию*. Например,

СТЕК	ВХОД
5	DUP
5 5	

Итак, SQR - это просто

```
        SQR  (n-n*n)
    DUP * ;
```

где операция "*" имеет в распоряжении две копии аргумента.

Со словом DUP в одну группу входят следующие слова:

DROP (удаляющее элемент с верхушки стека),

SWAP (меняющее местами два верхние элемента стека)

OVER (создающее копию элемента, следующего за верхним)

ROT (кладущее третий от верхушки элемент на стек)

и некоторые другие. Слова этой группы производят действие, подобное действию местоимений (этот, тот, другой и т. д.) в том смысле, что они позволяют ссылаться на аргументы, получаемые в разных частях предложения, посредством позиции, а не по имени. В качестве упражнения проверьте, что следующее Forth-выражение вычисляет функцию $y(m,n)=(m+n)(m-n)$

```
( m n )  OVER  OVER  + ROT ROT - *  (y)
```

(где комментарий подсказывает, что на стеке до выполнения функции и что после).

А. 12. Выбор варианта

Управляющая конструкция, которая интенсивно используется при программировании CAM - это оператор выбора, позволяющий выбрать для исполнения одно из нескольких альтернативных действий. Предположим, что у нас есть три слова BEEP, HONK и WHISTLE; мы можем создать новое слово SOUND, которое будет брать целочисленный аргумент с вершины стека - 0, 1, 2 и соответственно производить сигнал "бип", "гонг" или "свисток":

```
                    : SOUND (–)
{ BEEP HONK WHISTLE } ;
```

т. е. O SOUND исполнит BEEP , 1 SOUND - HONK и т. е. Список для выбора может состоять из любого числа *n* элементов; этому списку ставится в соответствие отрезок натурального ряда чисел, начиная с 0 и кончая и-1. Список обрамляется парой слов: "{" и "}". При попытке исполнить оператор выбора с аргументом, меньшим 0 или большим л-1, будет выдано сообщение об ошибке.

Предположим, вы хотите создать слово, возвращающее число дней в некотором месяце. Вам, вероятно, необходимо попробовать следующее:

```
                    : DAYS (month – days)
                1 -
{ 31 28 31 30 31 30
  31 31 30 31 30 31 } ;
```

Если положить, что месяцы нумеруются начиная с 1, то нужно вычесть 1 из номера месяца, чтобы удовлетворить требованиям оператора выбора, которому нужна последовательность 0, 1, ..., 11. Все это хорошо, но существует одна тонкость: в CAM Forth оператор выбора считает, что элементы списка выбора - это конкретные имена словаря; за небольшим исключением (упомянутым в разд. А.4.) числа *не* являются элементами словаря. Существует простой выход из этой ситуации: *перед* тем как определить слово DAYS, нужно занести числа в словарь в виде *констант*

```
28   CONSTANT  28   (-28)
30   CONSTANT  30   (-30)
31   CONSTANT  31   (-31)
```

С этого момента фрагмент 28 будет рассматриваться как слово, которое оставляет в стеке 28 (так же, как до этого делало *число* 28). Оператор выбора теперь будет считать 28 элементом списка.

А. 13. Условные предложения

Фраза между парой BEGIN и AGAIN находится в бесконечном цикле, та же фраза, но между DO и LOOP, повторяется столько раз, сколько определено двумя аргументами, которые DO берет из стека. Фраза между словами IF И THEN будет выполнена, только если аргумент, обнаруженный словом IF в стеке, равен логическому значению "истина"[1]. Слово

```
=     ( r a n - -  F|T)
```

сравнивает два аргумента m и п и возвращает логический "флаг", имеющий значение "истина", если аргументы равны, и "ложь" - в противном случае. Поэтому последующее слово произведет звуковой сигнал, только когда два верхних элемента стека равны

```
            : BEEP-IF-EQUAL  ( m n —)
          = IF
  BEEP THEN ;
```

Расширенная конструкция IF/THEN/ELSE используется так:

```
            : BEEP-OR-WHISTLE
          = IF
     BEEP ELSE
  WHISTLE THEN
HONK HONK HONK ;
```

Это слово произведет сигнал "бип", если *т и п* равны, и "свисток", если нет; после этого три раза прозвучит сигнал гонга.

Существует также *оператор цикла с условием,* имеющий форму "BEGIN ... UNTIL", в котором цикл исполняется до тех пор, пока значение логического флага не станет равным "истина", а также оператор вида "BEGIN ... WHILE ... REPEAT", в котором цикл повторяется до тех пор, пока значение логического флага остается равным "истина".

А.14. Логические выражения

При определении правила CAM иногда удобно рассматривать содержимое клеток CAM как логические величины ("включено", "выключено", "истина", "ложь"), а иногда как числа (О или 1 - или даже 0, 1, 2, Ъ{ когда речь идет о двух

[1] Как кодируются в Forth логические величины, пока не имеет значения; это будет обсуждено в следующем разделе.

плоскостях битов). Для того, чтобы правильно понимать, что одно слово передает через стек другому слову, важно знать соглашения, принятые в CAM Forth, касающиеся арифметических и логических выражений.

Этот многословный раздел можно считать руководством в тех случаях, когда возникают сомнения.

Как мы уже знаем, каждая клетка Forth представляет собой 16-битовый паттерн, который может по-разному трактоваться в зависимости от принятых соглашений. Например, его можно использовать для моделирования целых чисел от 0 до 65.535 ("числа без знака"), для целых от -32 768 до 32 767 ("числа со знаком"), для *символа* ASCII (использующего только 8 младших битов) и т.д. Такая клетка не несет в себе метки используемого типа кодирования: это задача программиста - сделать все так, чтобы "пользователь" структуры знал, какие соглашения использовать для ее интерпретации.

Предположим, что верхняя ячейка стека содержит паттерн 1000000000101010; три следующих слова ".", и., и EMIT напечатают на экране содержимое этой ячейки. Однако "." сочтет его числом со знаком и напечатает "-32726"; и. сочтет его числом без знака и напечатает "32810"; EMIT будет рассматривать его как символ и напечатает "*" (поскольку младшие 8 битов 00101010 представляют в коде ASCII символ "*").

Во многих случаях удобно рассматривать содержимое ячейки как набор отдельных битов - каждый представляет бинарный исход. Слова

NOT (p - г)
AND (p q - г)
OR (p q - г)
XOR (p q - г)

с этой точки зрения очень полезны, так как позволяют манипулировать битами по отдельности или целиком. Например, мы можем "выключить" 8 старших битов содержимого путем выполнения операции AND С соответствующей маской, а именно, 0000000011111111, в которой старшие 8 битов "выключены", а младшие 8 - "включены". Напомним определение логических операций, которые имеют вид

NOT	AND	OR	XOR
l-i-»• 1	**00 ^ 0**	00 ^ 0	00 ^ **0**
^*0* ,	**01,*t*-* 0** ,	**pi *U* 1** ,	01H-> **1** .
	10 ^ 0	**10 ^ 1**	**10 ^ 1**
	11 (-4 1	11 K-> **1**	**11 ^ 0**

В частности, логическое NOT дополняет до 2 свой однобитовый аргумент, т. е. слово Forth NOT образует дополнение каждого из 16 битов клетки. Остальные три логические операции действуют на соответствующие биты двух входных клеток и дают 16-битовый результат.

Чтобы направить условное предложение по тому или иному из двух возможных путей (см. предыдущий раздел), нужен двоичный "флаг" со значениями "истина" и "ложь".[1] Для этой цели было бы достаточно однобитового фрагмента; но клетки Forth имеют стандартный размер - 16 бит, и необходимо иметь соглашение: в каких случаях 16-битовое содержимое считать "истиной", а в каких - "ложью". Стандарт Forth-83 оговаривает, что слова, *возвращающие* логический флаг (такие как "=" и аналогичные "слова-сравнения") не кладут ничего в стек, кроме 1111111111111111 для "истины" или 0000000000000000 для "лжи". Для удобства эти величины занесены в словарь как coNSTANT-слова под именами TRUE и FALSE.[2] С другой стороны, слова, *ожидающие* логического флага (такие как IF И аналогичные "слова-условия"), будут считать значение 0000000000000000 "ложью", а *любое другое* значение - "истиной".

Если в качестве логических флагов "истина" и "ложь" используются значения слов TRUE и FALSE, то их в качестве флагов могут употреблять логические операции над битами NOT, AND и др. Однако при попытке неразборчиво воспользоваться более широким охватом оператора IF^ возникают некоторые тонкости. Мы приведем лишь один пример в качестве предостережения опрометчивому программисту.

Рассмотрим слово

```
: BEEP-IF-NOT-EQUAL ( и л - ]
= NOT I F
BEEP THEN ;
```

(см. предыдущий раздел), которое выдаст звуковой сигнал тогда и только тогда, когда *m~n*. Это слово будет работать так же, если заменить "=NOT" на "-". Действительно, если *m = n,* то их разность будет равна 0, что означает "ложь" для IF: с другой стороны, если *m* не равно и, то значение разности *m-n* будет содержать по крайней мере один ненулевой бит, что означает "истина" для IF.

[1] Когда существует более двух возможных исходов, обычно естественнее использовать оператор выбора (см. разд. АЛ 2.), чем многократно вложенные операторы IF .

[2] Заметим, что при печати^ чисел со знаком слово TRUE порождает -1, а слово FALSE порождает 0: при печати чисел без знака слово TRUE порождает 65.535 (FFFF в 16-ричной системе счисления).

[3] Например, применяя арифметику вместо логики для упрощения жизни.

Итак, если "-" работает "так же", как "=NOT", то будет ли "- NOT" работать так же, как и "=" в BEEP-IF-EQUAL предыдущего раздела? Если $m = n$, то их разность равна 0, а ее дополнение, образованное операцией NOT, является паттерном из одних единиц (TRUE-значение) - которое, конечно, воспринимается IF как "истина", как мы того и ожидали. Если, же *m^n,* то значение разности будет содержать несколько единиц, но может содержать и нули: поэтому дополнение, образованное операцией NOT, будет содержать единицы в некоторых позициях. И в этом случае оператор IF воспринимает его как "истина". Но этого мы уж никак не ожидали.

Так как слова, характеризующие отношения соседства - CENTER, NORTH и т.д. - возвращают 1 и о в качестве значения, то мы будем, где это целесообразно, использовать их в качестве "истина" и "ложь" соответственно. Логические операции, использующие такие однобитовые флаги, работают хорошо, кроме операции NOT (которая образует дополнение от всех 16 битов). Для получения дополнения от одного бита может использоваться последовательность "1 XOR". Напротив, операции сравнения - "=", ">", "<" и "<>" (не равно) могут быть использованы для преобразования однобитовых флагов в стандартные логические флаги.[1]

Наконец, полезно помнить, что слова >PLNO и AUXO, которые берут верхушку стека и помещают ее в виде элемента в справочную таблицу, используют младший бит значения верхушки стека, все остальные биты просто игнорируются ("объединенные" версии этих слов - PLNA и >AUXA - используют *два* младших бита).

А. 15. Литература для дальнейшего изучения Forth

Книга *"Starting Forth"* Лео Броди [8] - замечательное практическое введение в Forth. А в книге *"Thinking Forth"* того же автора [9] обсуждается методология, на которой зиждется язык Forth. В книге *"Inside Forth 83"* К. X. Тинга [54] дается тщательное описание внутренней структуры стандарта F83.

Периодическое издание *FORTH Dimentions,* публикуемое инициативной группой Forth - хороший источник новостей, приложений, технологий программирования, списков литературы, разработок в области программного и аппаратного обеспечения. Более академическим изданием является *Journal of Forth Application and Research,* публикуемый корпорацией Institute for Applied Forth Research, Inc.

[1] Слова **"0=", "0>"** являются аббревиатурой слов "О = " и т. д.

Приложение В

ОСНОВЫ АРХИТЕКТУРЫ САМ

Все, о чем пойдет речь далее, применимо с незначительными изменениями как к машине САМ-6, которая используется нами для примеров, так и для САМ-7, превосходящую САМ-6 размерами в две тысячи раз, разработка которой близка к завершению.

Хотя наиболее естественной архитектурой для машины клеточных автоматов могла бы быть архитектура полностью параллельного массива простых процессоров, однако такой подход представляет определенные технические трудности, в частности, когда приходится обдумывать расположение в трехмерном пространстве проводов, соединяющих огромное количество таких процессоров.

Архитектура САМ сохраняет основной концептуальный подход полностью параллельной машины, но с определенными изменениями, которые ведут к более практичной и экономичной реализации и к лучшему использованию современных технических возможностей.

Что касается параллельного подхода, то данная архитектура основывается на понятии *плоскостных модулей,* произвольное количество которых может быть соединено параллельно; каждый модуль - это плоскость битов, содержащая большое число точек (клеток). Однако, взятый отдельно, плоскостной модуль - это *конвейерный,* а не *параллельный* процессор.

В.1. Плоскостной модуль

В настоящем обсуждении мы ограничим наше внимание двумерным клеточным автоматом с одним битом данных на каждую его клетку. Автоматы большей размерности и с большим набором состояний будут рассмотрены в последующих разделах.

Весь массив разбит на прямоугольные части одинакового размера - *секторы.* Каждому сектору назначается свое аппаратное устройство - плоскостной модуль. Каждая плоскостной модуль состоит из трех основных секций: память для *переменных состояния,* секция *маршрутизации данных,* секция *функции перехода.*

Память хранит переменные состояния соответствующего сектора. Чтобы совершить один шаг обновления в этой области, необходимо последовательно прочитать текущие значения переменных состояния ^л ввести их в секцию маршрутизации

данных. Соответствующие новые значения, определенные справочной таблицей, возвращаются секцией маршрутизации в той же последовательности и снова записываются в память.

Из упомянутого выше последовательного потока данных секция маршрутизации извлекает в соответствии с подходящим упорядочиванием по времени девять величин, связанных в каждый момент с девятью соседними позициями клетки[1]: самой клетки - *Center,* ее четырех ближайших соседей - *North, South, East, West,* и четырех диагональных соседей - *N.East, N.West, S.East, S.West.* Секция маршрутизации данных обеспечивает также подходящую буферизацию, чтобы сделать обновление клеток выглядящим *синхронным,* хотя оно в действительности реализуется последовательным образом, и чтобы получить правильное вертикальное и горизонтальное "обертывание".

Желаемый набор из девяти сигналов (возможно, дополненный сигналами от других плоскостных модулей, см. разд. В.3 и В.4), параллельно передают в качестве аргументов в секцию функции перехода, которая использует справочную таблицу, чтобы вычислить соответствующее новое значение центральной клетки. После краткого путешествия через секцию маршрутизации данных новое значение передается в память, где оно заменяет текущее значение центральной клетки.

Заметим, что все вспомогательные процедуры, такие например, как подбор аргументов, исполняет секция маршрутизации; поэтому справочная таблица, которая является самым критичным ресурсом при моделировании, используется в полной мере. Более того, поскольку каждую справочную таблицу совместно использует большое число клеток, то становится практичным применять большую справочную таблицу, выполняя таким образом значительный объем вычислений за один шаг.

В.2. Большие массивы: склеивание границ

Произвольно большие двумерные массивы можно получить путем склеивания секторов встык по границам, т. е. путем обмена между конвейерными механизмами двух смежных секторов данными о тех клетках, которые принадлежат одному плоскостному модулю и соседствуют с клетками другого плоскост-

[1] В САМ-7 эта часть работы секции маршрутизации в значительной степени устранена, поскольку разбиение, которое базируется на относительном смещении точек начала координат плоскостей (см. разд. 15.6), образует основу для подбора соседей.

ного модуля. Размер плоскостного модуля САМ-6 - 256x256 клеток. В полностью параллельной архитектуре это привело бы к появлению плоскостного модуля с тысячами внешних выводов; в конвейерной архитектуре обмен информацией на границах последовательный, и достаточно четырех линий двусторонней связи, соответствующих четырем смежным секторам.[1]

Склеивая секторы таким способом, можно получить произвольно большой *лист;* в большинстве случаев такой лист будет обернутым, т. е. верхняя граница будет соединена с нижней, а левая - с правой. Таким образом, глобальная топология листа будет топологией тора. Ту же процедуру склеивания используют как для расширения массива, так и для удаления границ путем обертывания.

Заметим, что склеивание плоскостных модулей выполняется один раз на стадии маршрутизации данных. Поэтому как с логической, так и с физической точек зрения секция функции перехода *совершенно не зависит* от ряда деталей реализации: (а) того факта, что лист состоит из склеенных плоскостных модулей, (Ь) что хранение и маршрутизация производятся на двумерной основе и для каждой плоскости битов независимо, (с) что операции выполняются конвейерным образом.

В.3. Увеличение числа состояний клетки: группирование листов

Когда получены листы желаемого размера, то дальнейшую настройка аппаратной части машины клеточных автоматов производят путем выбора подходящих сигналов в качестве аргументов функции перехода. В частности, для того, чтобы получить больший набор состояний клетки, достаточно *сгруппировать* ряд листов, т. е. взять в качестве входов в справочную таблицу каждого листа выборку выходов соседей, принадлежащих другим листам группы. Подобным образом сгруппированный ряд образует *слой* клеточного автомата, содержащий полностью сформированную клетку.

[1] Такой способ объединения секторов опирается на тот факт, что клеточная память отдельного плоскостного модуля логически обернута: клетка на физической границе сектора рассматривает клетки как на этой, так и на другой границе в качестве своих соседей. Поскольку схема сканирования при обновлении клеток одна и та же для всех плоскостных модулей, то у всех у них подлежащие обмену граничные соседи становятся доступны одновременно. Обменивая конвейеры, а не соседей, мы можем добиться того же результата, используя одну связь с соседними секторами *независимо* от размера окрестности.

В.4. Увеличение размерности: наложение слоев

Теперь слои можно *сложить в стопку,* один поверх другого, путем объединения в качестве входов функции перехода каждого слоя выборку соседних выходов слоев, лежащих непосредственно ниже и выше данного. Это возможно, поскольку все плоскостные модули будут обновлять соответствующие клетки одновременно. Таким путем мы сможем объединить машины типа САМ в трехмерный клеточный автомат. Это построение можно продолжить, чтобы получить клеточные автоматы размерностью четыре или больше.

В.5. Отображение и анализ

Каждый из четырех плоскостных модулей машины САМ-6 порождает данные со скоростью около 6 Мбит/сек. Если для отображения необходимо существенное преобразовать эту информацию, то это потребует ресурсов того же порядка величины, как и ее получение.

В конвейерной архитектуре сканирование массива осуществляется последовательно; при соответствующем выборе параметров сканирования эта информация может быть выведена в виде, пригодном для отображения на растровом сканирующем устройстве. В плоскостном модуле САМ число строк и столбцов массива, покрываемых данным модулем, последовательность сканирования и схема синхронизации таковы, что отвод от конвейера может непосредственно питать черно-белый CRT-монитор. Конечно, выходы ряда сгруппированных плоскостных модулей (см. разд. В3), которые вместе представляют значение многобитовой переменной состояния, могут быть объединены в RGB-сигнал и отображены на одном *цветном* мониторе.[1]

Преимущества такой организации не ограничиваются непосредственным отображением. В плоскостном модуле всю информацию о соседях, потенциально доступную для функции перехода, легко собрать и направить в дополнительную справочную таблицу. Таким путем можно вычислить и послать на экран произвольную *выходную* функцию, а не только значение текущей центральной клетки. Это позволяет выполнять "на ходу" предварительную графическую обработку достаточно большого объема (такой подход напоминает методы окрашивания, используемые в микроскопии для выявления тех или иных структур в исследуемой ткани).

[1] Когда отображение не нужно, часы САМ можно пустить быстрее, чем частота видеосигнала, чтобы, например, более часто обновлять массив меньших размеров, скажем, со скоростью несколько тысяч кадров в секунду.

Далее, поток значений выходной функции можно представить в виде гистограммы, накопить и сравнить с заранее выбранными пороговыми значениями и вообще использовать для обработки в реальном времени и управления динамикой системы. В частности, можно выявить и подсчитать встречаемость любого выделеного локального паттерна.

Наконец, поскольку на любом шаге обновления поток всех данных в каждом плоскостном модуле направляют через конвейерную магистраль, то достаточно единственного двустороннего отвода на этой магистрали для обеспечения любому внешнему устройству доступа ко всему объему данных как по чтению, так и по записи. Совокупность таких отводов - по одному на каждый плоскостной модуль - представляет собой чрезвычайно высокоскоростную шину (в САМ-7, в которой планируется наличие 1024 плоскостных модулей с 512x512 клетками каждая, мы получим суммарную длину слова 1024 бит и синхронную скорость слов 40 нсек.), через которую экспериментатор имеет постоянный доступ к полному состоянию системы в процессе ее эволюции. Такая "шина-маховик" - уникальная особенность архитектуры САМ.

Отметим в заключение, что конвейерная магистраль, питаемая в соответствии с хорошо подобранным форматом последовательности сигналов и снабженная правильно размещенными отводами, образует шину общего назначения, на которую можно "навесить" не только функцию перехода, но также огромное разнообразие функций отображения, анализа и управления, не отягощая при этом накладными расходами процесс моделирования. Как и в физическом эксперименте, любая часть системы потенциально доступна для интерактивного воздействия и измерения.

В.6. Модульность *и* расширяемость

В отличие от других современных схем параллельных вычислений, архитектура САМ действительно не зависит от размеров машины, и очень большую машину клеточных автоматов можно построить простым соединением нужного числа плоскостных модулей. Ограничения устанавливаются требованиями экономики, а не электротехники или логического проектирования. Поскольку не существует "адресов" в традиционном смысле, то область данных не ограничена размером адресного слова. Единственный временной сигнал, который нужно подать на все блоки плоскостных модулей, - это последовательность тактовых импульсов, и, поскольку сигналы можно заново синхронизировать в каждом блоке, то система может обойтись резервом времени между блоками, сравнимым с шириной тактов импульса.

ЛИТЕРАТУРА

[1]*ALADYEV, Viktor, "Computability in Homogeneous Structures," *Izv. Akad. Nauk. Estonian SSR, Fiz.-Mat.* 21 (1972), 80-83.

[2] AMOROSO, Serafino, and Y. N. PATT, "Decision Procedures for Surjectivity and Injectivity of Parallel Maps for Tessellation Structures," J. *Comp. Syst. Sci.* 10 (1975), 77-82.

[3] BANKS, Edwin, "Information Processing and Transmission in Cellular Automata," *Tech. Rep. MAC TR-81,* MIT Project MAC (1971)

[4] BENNETT, Charles, "Logical Reversibility of Computation," *IBM* J. *Res. Develop.* 6 (1973), 525-532.

[5] BENNETT, Charles, and Geoff GRINSTEIN, "Role of Irreversibility in Stabilizing Complex and Nonenergodic Behavior in Locally Interacting Discrete Systems," *Phys. Rev. Lett.* 55 (1985), 657-660.

[6] BERLEKAMP, Elwyn, John CONWAY, and Richard GUY, *Winning ways for your mathematical plays,* vol. 2, Academic Press (1982).

[7] BRENDER, Ronald, "A Programming System for the Simulation of Cellular Spaces," *Tech. Rep. 25,* CONCOMP, The Univ. of Michigan (1970).

[8] BRODIE, Leo, *Starting FORTH,* Prentice Hall (1981).

[9] BRODIE, Leo, *Thinking FORTH,* Prentice Hall (1984).

[10] BURKS, Arthur (ed.), *Essays on Cellular Automata,* Univ. 111. Press (1970).

[11] CODD, E. F., *Cellular Automata,* Academic Press (1968).

[12] COX, J. Theodore, David GRIFFEATH, "Recent results for the stepping stone model," *University of Wisconsin Math Department preprint.*

[13] CREUTZ, Michael, "Deterministic Ising Dynamics," *Annals of Physics* **167** (1986), 62-76.

[14] D'HUMIERES, Dominique, Pierre LALLEMAND, and T. SHIMOMURA, "Lattice Gas Cellular Automata, a New Experimental Tool for Hydrodynamics," Preprint LA-UR-85-4051, Los Alamos National Laboratory (1985).

[15] FARMER, Doyne, Tommaso TOFFOLI, and Stephen WOLFRAM (eds.), *Cellular* Automata, North-Holland (1984).

[16]*FELLER, William, *An Introduction to Probability Theory and Its Applications,* vol. I, 3rd ed., Wiley (1968).

[17] FREDKIN, Edward, and Tommaso TOFFOLI, "Conservative Logic," *Int. J. Theor. Phys.* 21 (1982), 219-253.

[18] FRISCH, Uriel, Brosl HASSLACHER, and Yves POMEAU, "Lattice-Gas Automata for the Navier-Stokes Equation," *Phys. Rev. Lett.* 56 (1986), 1505-1508.

[19] GACS, Peter, and John REIF, *Proc. 17-th ACM Symp. Theory of Computing* (1985), 388-395.

[20] GARDNER, Martin, "The Fantastic Combinations of John Conway's New Solitaire Game 'Life'," *Sc. Am.* 223:4 (April 1970), 120-123.

[21] GREENBERG, J., and S. HASTINGS, "Spatial Patterns for Discrete Models of Diffusion in Excitable Media," *SIAM J. Appl. Math.* 34 (1978), 515.

[22] HAYES, Brian, "The cellular automaton offers a model of the world and a world unto itself," *Scientific American* 250:3 (1984), 12-21.

[23] HARDY, J., O. DE PAZZIS, and Yves POMEAU, "Molecular dynamics of a classical lattice gas: Transport properties and time correlation functions," *Phys. Rev.* A13 (1976), 1949-1960.

[24] HEDLUND, G. A., K. I. APPEL, and L. R. WELCH, "All Onto Functions of Span Less Than or Equal To Five," Communications Research Division, working paper (July 1963).

[25] HEDLUND, G. A., "Endomorphism and Automorphism of the Shift Dynamical System," *Math. Syst. Theory* 3 (1969), 51-59.

[26] HERRMANN, Hans, "Fast algorithm for the simulation of Ising models," Saclay preprint no. 86-060 (1986).

[27] HOLLAND, John, "Universal Spaces: A Basis for Studies in Adaptation," *Automata Theory,* Academic Press (1966), 218-230.

[28] KADANOFF, Leo, "On two levels," *Physics Today* 39:9 (September 1986), 7-9.

[29] KIMURA, M., G. WEISS, "The stepping stone model of population structure and the decrease of genetic correlation with distance," *Genetics* 49 (1964), 561-576.

[30] KIRKPATRICK, Scott, C D. GELATT Jr., M.P. VECCHI, "Optimization by Simulated Annealing," Science 220 (1983), 671-680.

[31]*KNUTH, Donald, The *Art of Computer Programming,* vol. 2, Seminumerical Algorithms, 2nd ed., Addison-Wesley (1981).

[32] LANDAUER, Rolf, "Irreversibility and heat generation in the computing process," *IBM J. Res. Devel.* 5 (1961), 183-191.

[33] LANDAU, L., E. LIFSHITZ, *Mechanics,* Pergamon Press (1960).

[34] MANDELBROT, Benoit, *The Fractal Geometry of Nature,* W. H. Freeman (1982).

[35] MARGOLUS, Norman "Physics-like models of computation," *Physica* 10D (1984), 81-95.

[36] MARGOLUS, Norman, Tommaso TOFFOLI, and Gerard VICHNIAC, "Cellular-Automata Supercomputers for Fluid Dynamics Modeling," *Phys. Rev. Lett.* 56 (1986), 1694-1696.

[37] MARGOLUS, Norman, "Quantum Computation," Proceedings of a conference on New Ideas and Techniques in Quantum Measurement Theory (December 1985), to be published in the *Annals of the New York Academy of Sciences* (1986).

[38] MARGOLUS, Norman, "Partitioning Cellular Automata," in preparation.

[39] MARUOKA, Akira, and Masayuki KIMURA, "Conditions for Injectivity of Global Maps for Tessellation Automata," *Info. Control* 32 (1976), 158-162.

[40] MARUOKA, Akira, and Masayuki KIMURA, "Injectivity and Surjectivity of Parallel Maps for Cellular Automata," J. *Comp. Syst. Sci.* 18 (1979), 47-64.

[41] MEZARD, M., "On the Statistical Physics of Spin Glasses," *Disordered Systems and Biological Organization* (E. BIENENSTOCK et al., ed.), Springer-Verlag (1986), 119-132.

[42] ORSZAG, Steven, and Victor YAKHOT, "Reynolds Numbers Scaling of Cellular-Automaton Hydrodynamics," *Phys. Rev. Lett.* 56 (1986), 1691-1693.

[43] PACKARD, Norman, and Stephen WOLFRAM, "Two-dimensional cellular automata," *J. Stat. Phys.* 38 (1985), 901-946.

[44] PEARSON, Robert, "An Algorithm for Pseudo Random Number Generation Suitable for Large Scale Integration," J. *Computat. Phys.* 3 (1983), 478-489.

[45] POMEAU, Yves, "Invariant in Cellular Automata," *J. Phys.* **A17** (1984), L415-L418.

[46] PRESTON, Kendall, and Michael DUFF, *Modem Cellular Automata, Theory and Applications,* Plenum Press (1984).

[47] REITER, Carla, "Life and death on a computer screen," *Discover* (August 1984), 81-83.

[48] RICHARDSON, D., "Tessellation with Local Transformations," *J. Comp. Syst. Sci.* 6 (1972), 373-388.

[49] ROSENBERG, L, "Spin Glass and Pseudo-Boolean Optimization," *Disordered Systems and Biological Organization* (E. BIENENSTOCK et al., ed.), Springer-Verlag (1986), 327-331.

[50] SALEM, James, and Stephen WOLFRAM, "Thermodynamics and Hydrodynamics of Cellular Automata," *Theory and Applications of Cellular Automata* (Stephen WOLFRAM ed.), World Scientific (1986), 362-366.

[51] SANDER, Leonard, "Fractal growth processes," Nature **322** (1986) 789-793.

[52] SMITH, Alvy, "Cellular Automata Theory," *Tech. Rep. 2,* Stanford Electronic Lab., Stanford Univ. (1969).

[53] STANLEY, H. Eugene, and Nicole OSTROWSKY, *On Growth and Form,* Martinus Nijhoff (1986).

[54] TING, C H., Inside *F83,* Offete Press, 1306 South B. St., San Mateo, CA 94402.

[55] TOFFOLI, Tommaso, "Cellular Automata Mechanics'," *Tech. Rep. 208,* Сотр. Сотт. Sci. Dept., The Univ. of Michigan (1977).

156] TOFFOLI, Tommaso, "Integration of the Phase-Difference Relations in Asynchronous Sequential Networks," Automata, *Languages, and Programming* (Giorgio AUSIELLO and Corrado BOHM ed.), Springer-Verlag (1978), 457-463.

[57] TOFFOLI, Tommaso, "Bicontinuous extension of reversible combinatorial functions," Maths. *Syst. Theory* 14 (1981), 13-23.

[58] TOFFOLI, Tommaso, "Reversible Computing," *Automata, Languages and Programming* (DE BARKER and VAN LEEUWEN eds.), Springer-Verlag (1980), 632-644.

[59] TOFFOLI, Tommaso, "CAM: A high-performance cellular-automaton machine," *Physica* **10D** (1984), 195-204.

[60] TOFFOLI, Tommaso, and Norman MARGOLUS, "The CAM-7 Multiprocessor: A Cellular Automata Machine," *Tech. Memo LCS-TM-289,* MIT Lab. for Сотр. Sci. (1985).

[61] TOFFOLI, Tommaso, "Cellular automata as an alternative to (rather than an approximation of) differential equations in modeling physics," *Physica* **10D** (1984), 117-127.

[62] TOFFOLI, Tommaso, and Norman MARGOLUS, *Invertible Cellular Automata,* in preparation.

[63] TUCKER, Jonathan, "Cellular automata machine: the ultimate parallel computer," *High Technology* 4:6 (1984), 85-87.

[64] ULAM, Stanislaw, "Random Processes and Transformations," *Proc. Int. Congr. Mathem.* (held in 1950) 2 (1952), 264-275.

[65]*VAN DYKE, Milton, *An Album of Fluid Motion,* Parabolic Press (1982).

[66] VICHNIAC, Gerard, "Simulating physics with cellular automata," Physica **10D** (1984), 96-115.

[67] VICHNIAC, Gerard, "Cellular automata models of disorder and organization," *Disordered Systems and Biological Organization* (BIENENSTOCK et al. eds.), Springer-Verlag (1986), 1-20.

[68]*VON NEUMANN, John, *Theory of Self-Reproducing Automata* (edited and completed by Arthur BURKS), Univ. of Illinois Press (1966).

[69] WITTEN, Thomas, and Leonard SANDER, *Phys. Rev. Lett.* 47 (1981), 1400.

[70] WOLFRAM, Stephen, "Statistical mechanics of cellular automata," Rev. Mod. *Phys.* **55** (1983), 601-644.

[71] WOLFRAM, Stephen, "Universality and Complexity in Cellular Automata," *Physica* **10D** (1984), 1-35.

[72] WOLFRAM, Stephen, "Computation Theory of Cellular Automata," *Commun. Math. Phys.* 96 (1984), 15-57.

[73] WOLFRAM, Stephen, "Random-Sequence Generation by Cellular Automata," *Adv. Applied Math. 7* (1986), 123-169.

[74] WOLFRAM, Stephen (ed.), *Theory and Applications of Cellular Automata,* World Scientific (1986).

[75] ZAIKIN, A., and A. ZHABOTINSKY, Nature **225** (1970), 535.

[76] ZUSE, Konrad, Reclmender Raum, Vieweg, Braunschweig (1969); translated as "Calculating Space," *Tech. Transl. AZT-70-164-GEMIT,* MIT Project MAC (1970).

Звездочкой * отмечены работы, имеющиеся на русском языке:

*[1] Аладьев В. Известия АН Эст. ССР, Физика и Математика, 21(1972), 80-83.

*[16] Феллер В. Введение в теорию вероятностей и ее применения: В 2-х томах, т. 1: Пер. с англ. - М.: Мир, 1983.

*[31] Кнут Д. Искусство программирования для ЭВМ: В 3-х томах: т.2: Получисленные алгоритмы: Пер. с англ. - М.: Мир, 1977.

*[65] Ван-Дайк М. (ред) Альбом течений жидкости и газа: Пер. с англ. - М.: Мир, 1986.

*[68] Фон Нейман Дж. Теория самовоспроизводящихся автоматов: Пер. с англ. - М.: Мир, 1971.

ОГЛАВЛЕНИЕ

УВАЖАЕМЫЙ ЧИТАТЕЛЬ!

Ваши замечания о содержании книги,
ее оформлении,
качестве перевода и другие
просим присылать по адресу:
129820, 1-й Рижский пер., 2, издательство "Мир".

Научное издание

Томмазо Тоффоли, Норман Марголус

МАШИНЫ КЛЕТОЧНЫХ АВТОМАТОВ

Зав. редакцией чл.-корр. АН СССР В. И. Арнольд
Зам. зав. редакцией А. С. Попов
Научный редактор С. В. Чудов
Художник Ю. С. Урманчиев
Художественный редактор В. И. Шаповалов
Корректор Е. В. Морозова
Технический редактор Т. А. Мирошина

ИБ № 7315

Оригинал-макет подготовлен
на персональном компьютере
и отпечатан на лазерном принтере
в издательстве "Мир"

Подписано к печати 26.06.90. Формат 60x90 1/16
Бумага офсетная № 1. Печать офсетная.
Гарнитура Тайме. Объем 8.75 бум. л. Усл. печ. л. 17.50.
Усл.кр.-отт. 17.93. Уч.-изд. л. 15.52. Изд N 1/6767.
Тираж 5500 экз. Зак.1608. Цена 2 р. 50 к.

Издательство "Мир"
В/О "Совэкспорткнига"
Государственного комитета СССР по печати
129820, ГСП, Москва, И-ПО, 1-й Рижский пер., 2.

Можайский полиграфкомбинат В/О "Совэкспорткнига"
Государственного комитета СССР по печати
143200, г. Можайск, ул. Мира, 93

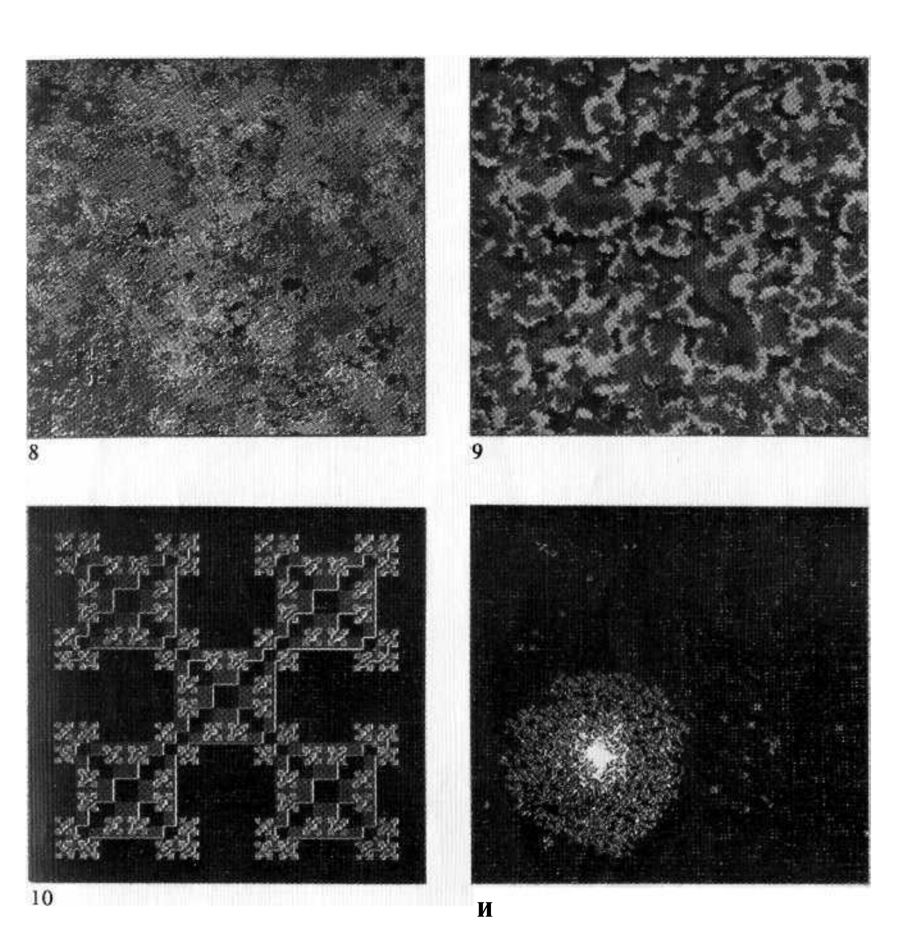


Фото 8. Случайное распространение четырех конкурирующих популяций служит моделью генетического дрейфа.

Фото 9. Другая модель генетического дрейфа, в которой конкуренция характеризуется циклической упорядоченностью относительной силы разных состояний (разд. 9.6). Подобные механизмы могут быть использованы для генерирования синхронизирующих сигналов, позволяющих асинхронной системе выполнять детерминированные вычисления.

Фото 10, 11. Фракталы (разд. 12.8.1) и самоорганизация (разд. 12.8.2) в обратимых системах.

2 р. 50 к.

12

13

14

15

Фото 12. Дендритообразный кристалл, полученный в результате ограниченного диффузией агрегирования (разд. 15.7) из одноклеточного зародыша.

Фото 13. Звуковые волны в решеточном газе (разд. 16.1). Обратите внимание на то, что форма распространяющегося возмущения является круговой, хотя индивидуальные частицы газа могут передвигаться лишь по вертикали и горизонтали.

Фото 14. Спиновая модель Изинга, использующая в качестве переменной состояния энергию связей, а не ориентацию спинов (разд. 17.7).

Фото 15. Обратимое правило второго порядка (TIME-TUNNEL, разд. 6.3). Начальное состояние показано на фото 2. Обратите внимание на сохранение некоторых структур.

ISBN 5-03-001619-8 (русск.)
ISBN 0-262-20060-0 (англ.)